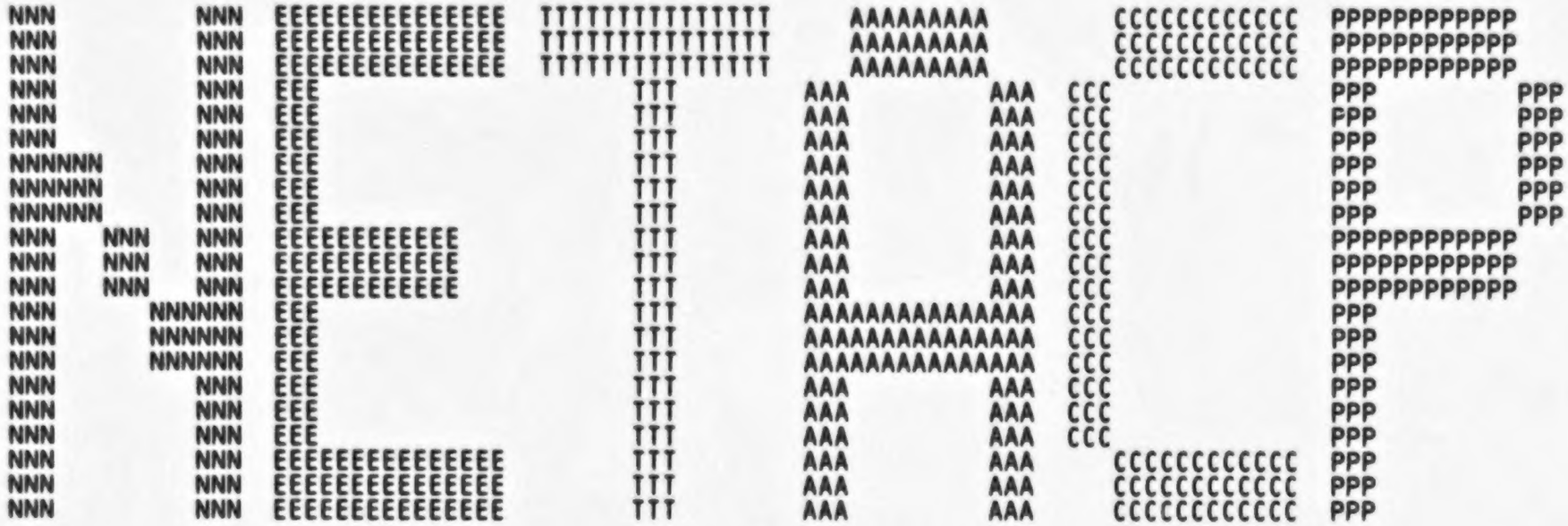
NNN NNN EEEEEEEEEEEEEEE TTTTTTTTTTTTTTT AAAAAAAAA CCCCCCCCCCCC PPPPPPPPPPPP
NNN NNN EEE TTT AAA AAA CCC PPP PPP
NNNNNN NNN EEE TTT AAA AAA CCC PPP PPP
NNN NNN NNN EEEEEEEEEEEE TTT AAA AAA CCC PPPPPPPPPPPP
NNN NNNNNN EEE TTT AAAAAAAAAAAAAAA CCC PPP
NNN NNN EEE TTT AAA AAA CCCCCCCCCCCC PPP

```
NN      NN  EEEEEEEEEE  TTTTTTTTTT    CCCCCCCC  NN      NN  FFFFFFFFFF
NN      NN  EEEEEEEEEE  TTTTTTTTTT    CCCCCCCC  NN      NN  FFFFFFFFFF
NN      NN  EE              TT      CC          NN      NN  FF
NN      NN  EE              TT      CC          NN      NN  FF
NNNN    NN  EE              TT      CC          NNNN    NN  FF
NNNN    NN  EE              TT      CC          NNNN    NN  FF
NN  NN  NN  EEEEEEEE        TT      CC          NN  NN  NN  FFFFFFFF
NN  NN  NN  EEEEEEEE        TT      CC          NN  NN  NN  FFFFFFFF
NN    NNNN  EE              TT      CC          NN    NNNN  FF
NN    NNNN  EE              TT      CC          NN    NNNN  FF
NN      NN  EE              TT      CC          NN      NN  FF              ....
NN      NN  EE              TT      CC          NN      NN  FF              ....
NN      NN  EEEEEEEEEE      TT        CCCCCCCC  NN      NN  FF              ....
NN      NN  EEEEEEEEEE      TT        CCCCCCCC  NN      NN  FF              ....

LL          IIIIII      SSSSSSSS
LL          IIIIII      SSSSSSSS
LL            II      SS
LL            II      SS
LL            II      SS
LL            II        SSSSSS
LL            II        SSSSSS
LL            II              SS
LL            II              SS
LL            II              SS
LL            II              SS
LLLLLLLLLL  IIIIII    SSSSSSSS
LLLLLLLLLL  IIIIII    SSSSSSSS
```

Table of contents

```
0000     1          .TITLE  NETCNF  - Configuration data base access routines
0000     2          .IDENT  'V04-000'
0000     3          .DEFAULT DISPLACEMENT,WORD
0000     4
0000     5  ;*****************************************************************************
0000     6  ;*                                                                           *
0000     7  ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                  *
0000     8  ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                   *
0000     9  ;*  ALL RIGHTS RESERVED.                                                     *
0000    10  ;*                                                                           *
0000    11  ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED     *
0000    12  ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE    *
0000    13  ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER     *
0000    14  ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY     *
0000    15  ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY     *
0000    16  ;*  TRANSFERRED.                                                             *
0000    17  ;*                                                                           *
0000    18  ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE     *
0000    19  ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT    *
0000    20  ;*  CORPORATION.                                                             *
0000    21  ;*                                                                           *
0000    22  ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS     *
0000    23  ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
0000    24  ;*                                                                           *
0000    25  ;*                                                                           *
0000    26  ;*****************************************************************************
0000    27  ;
0000    28  ;
0000    29  ; FACILITY:     NETWORK ACP
0000    30  ;
0000    31  ; ABSTRACT:
0000    32  ;               This module provides access to the NETACP configuration
0000    33  ;               database.
0000    34  ;
0000    35  ; ENVIRONMENT:
0000    36  ;               Kernel mode
0000    37  ;
0000    38  ; AUTHOR:       A.Eldridge      14-JAN-80
0000    39  ;
0000    40  ; MODIFIED BY:
0000    41  ;
0000    42  ;       V011    RNG0011         Rod Gamache             16-Mar-1984
0000    43  ;               Fix routine that calls action routines to not clobber the
0000    44  ;               return status in R0.
0000    45  ;
0000    46  ;       V010    RNG0010         Rod Gamache             7-Feb-1984
0000    47  ;               Fix return from GET_FIELD for register descriptor to be
0000    48  ;               zero on error returns.
0000    49  ;               Fix possible stack problem with CNF$DELETE routine.
0000    50  ;
0000    51  ;       V009    TMH0009         Tim Halvorsen           17-May-1983
0000    52  ;               Fix bug in GET_FIELD and COMPARE_ACT which assumes that
0000    53  ;               the field is a longword, and picks up the value before
0000    54  ;               it finds out it may be a "bit".  If the bit number is
0000    55  ;               high enough, this may cause a spurious reference off the
0000    56  ;               end of the structure, and if the next page is a null page,
0000    57  ;               the system will crash.
```

```
0000     58 ;
0000     59 ;       V008    RNG0008         Rod Gamache             29-Mar-1983
0000     60 ;               Add code to support binary balanced trees for the NDI
0000     61 ;               database.
0000     62 ;
0000     63 ;       V007    TMH0007         Tim Halvorsen           05-Nov-1982
0000     64 ;               Add concept of action routines which can both read and
0000     65 ;               write a parameter (in addition to the existing concept of
0000     66 ;               action routines which only read a parameter).
0000     67 ;
0000     68 ;       V006    TMH0006         Tim Halvorsen           02-Jul-1982
0000     69 ;               Modify routine which stores a string parameter when
0000     70 ;               one already exists, so that, if the string is equal
0000     71 ;               to, or less than the size of the original string, then
0000     72 ;               the space is simply reused, rather than returning
0000     73 ;               an error.  This is needed because NI datalink drivers
0000     74 ;               now deal more with string parameters (NI addresses).
0000     75 ;               Enhance CNF$VERIFY so that it properly detects a
0000     76 ;               parameter which is not in the semantic table, but
0000     77 ;               is within the range of allowable indicies (a hole
0000     78 ;               in the table).
0000     79 ;
0000     80 ;       V005    TMH0005         Tim Halvorsen           16-Jun-1982
0000     81 ;               Add code to handle new type of field access control
0000     82 ;               called "no external read or write access" (ACC_NE).
0000     83 ;               Add $DYNDEF definition.
0000     84 ;
0000     85 ;       V004    TMH0004         Tim Halvorsen           04-Apr-1982
0000     86 ;               Remove spurious instruction and label.
0000     87 ;               Special case NFB$C_WILDCARD as a search field ID in
0000     88 ;               KEY_SRCH, in order to remove extra code in CTLALL.
0000     89 ;               Replace call to NET$APPLY_DFLT with a call to a CNR
0000     90 ;               specific action routine to apply the default values.
0000     91 ;               Return BADPARAM from GET_DSC if read access not allowed,
0000     92 ;               rather than returning a zero.
0000     93 ;               Make CNF$INIT a local routine, since it is not called by
0000     94 ;               any other module.
0000     95 ;               Modify calling sequence to field action routines, so that
0000     96 ;               a scratch buffer is automatically allocated here before
0000     97 ;               calling the routine, to avoid the expense of having each
0000     98 ;               routine do it.  In addition, all registers are automatically
0000     99 ;               saved over an action routine call.
0000    100 ;               Remove CNF$GET_ADDR routine, as it is no longer called
0000    101 ;               by anyone as a result of the action routine changes.
0000    102 ;               Add routine to search given a list of search keys.
0000    103 ;               Remove code to support FNDNEXT operator.
0000    104 ;               Fix FNDMIN and FNDMAX support so that it correctly
0000    105 ;               returns the matched CNF in R10.
0000    106 ;               Rename CNF$T_MASK to CNF$L_MASK.
0000    107 ;               Rename CNR$T_SEM_TAB to CNR$L_SEM_TAB.
0000    108 ;               Make default word addressing mode and remove all
0000    109 ;               explicit addressing mode specifiers.
0000    110 ;               Use SETBIT and CLRBIT macros where ever possible.
0000    111 ;
0000    112 ;       V003    TMH0003         Tim Halvorsen           25-Mar-1982
0000    113 ;               Fix routine which compresses a CNF block to correctly
0000    114 ;               initialize the amount of space used for strings, to
```

```
0000   115 ;                   prevent a continual increase in the block size for
0000   116 ;                   each block compression.
0000   117 ;
0000   118 ;       V02-002 ADE0050         A.Eldridge                  19-Jan-1982
0000   119 ;                   Added call to NET$APPLY_DFLT which applies default values
0000   120 ;                   to selected CNF parameters when an entry is about to
0000   121 ;                   inserted into the database.
0000   122 ;
0000   123 ;       V02-001 ADE0007         A.Eldridge
0000   124 ;                   General cleanup.
0000   125 ;--
```

```
                        0000   127          .SBTTL  Declarations
                        0000   128 ;
                        0000   129 ; INCLUDE FILES:
                        0000   130 ;
                        0000   131          $DYNDEF                 ; Dynamic structure types
                        0000   132
                        0000   133          $CNRDEF                 ; Configuration Root Block
                        0000   134          $CNFDEF                 ; Configuration Data Block
                        0000   135          $NETSYMDEF              ; Miscellaneous symbol definitions
                        0000   136          $NFBDEF                 ; ACP control QIO definitions
                        0000   137
                        0000   138
                        0000   139 ;
                        0000   140 ; EQUATED SYMBOLS:
                        0000   141 ;
00000000                0000   142 STR_OFF = 0              ; String descriptor string self-relative offset
00000002                0000   143 STR_LNG = 2              ; String descriptor string size
                        0000   144
                        0000   145
0000044C                0000   146 TMP_LTH =  1100                      ; Length of temp buffer
                        0000   147
                        0000   148 ;
                        0000   149 ; OWN STORAGE
                        0000   150 ;
                        0000   151
                  00000000      152          .PSECT  NET_PURE,NOWRT,NOEXE,LONG
                        0000   153
                        0000   154
0000044C                0000   155 TMPBUF_DESC::    .LONG   TMP_LTH         ; Descriptor of TMP_BUF for external use
00000004'               0004   156                  .ADDRESS TMP_BUF
                        0008   157
                  00000000      158          .PSECT  NET_IMPURE,WRT,NOEXE
                        0000   159
00000004                0000   160 SELECT_CNF:      .BLKL   1               ; Currently selected min/max CNF
0000000C                0004   161 SELECT_VALUE:    .BLKL   2               ; Min/max value assoc. with SELECT_CNF
                        000C   162
                      00000C   163 TMP_B_FLAGS:     .BYTE   0               ; Buffer flags
00000000                000D   164 TMP_V_VAL = 0                            ; 1 if TMP_VAL in use, else 0
00000001                000D   165 TMP_V_BUF = 1                            ; 1 if buffer in use, else 0
                        000D   166
                  00000000      167          .PSECT  TABLES_IMPURE,WRT,NOEXE,GBL
                        0000   168
00000000                0000   169 TMP_VAL:          .LONG   0               ; Tmp storage for returned value
                        0004   170                                          ; and for "short" decriptor of TMP_BUF
                        0004   171                                          ; when returning strings
                        0004   172
00000450                0004   173 TMP_BUF:          .BLKB   TMP_LTH         ; Buffer for returning strings
                        0450   174 TMP_BUF_END:                             ; Address of first byte past buffer
00000000                0450   175                  .LONG   0               ; Leave an extra longword
                        0454   176
                  00000000      177          .PSECT  NET_CODE,NOWRT,EXE
```

```
                     0000   179          .SBTTL  CNF$PRE_SHOW - Pre-SHOW processing
                     0000   180 ;+
                     0000   181 ; CNF$PRE_SHOW  - Pre-process CNF for a "show" QIO
                     0000   182 ;
                     0000   183 ; Dispatch to database specific action routine to pre-process a CNF entry
                     0000   184 ; before a "show" QIO is processed for that entry.
                     0000   185 ;
                     0000   186 ; INPUTS:        R11     CNR pointer
                     0000   187 ;                 R10     CNF pointer
                     0000   188 ;                 R9-R7   Scratch
                     0000   189 ;                 R5-R0   Scratch
                     0000   190 ;
                     0000   191 ; OUTPUTS:       R11,R10 Preserved
                     0000   192 ;                 R6      Preserved
                     0000   193 ;
                     0000   194 ;                 All other regs are clobbered.
                     0000   195 ;-
                     0000   196 CNF$PRE_SHOW::                               ; "Show" QIO pre-processing
          56   DD    0000   197          PUSHL   R6                          ; Save reg
    1C BB    16    0002   198          JSB     @CNR$L_ACT_SHOW(R11)        ; Call action routine
       56 8ED0    0005   199          POPL    R6                          ; Restore reg
             05    0008   200          RSB                                 ; Done
```

```
                        0009   202          .SBTTL  CNF$PRE_QIO - Pre-QIO processing
                        0009   203 ;+
                        0009   204 ; CNF$PRE_QIO   - Pre-process database to prepare it for a QIO
                        0009   205 ;
                        0009   206 ; Dispatch to database specific action routine to pre-process a CNF entry
                        0009   207 ; before a "show" QIO is processed for that entry.
                        0009   208 ;
                        0009   209 ; INPUTS:       R11     CNR pointer
                        0009   210 ;
                        0009   211 ; OUTPUTS:      R11     Unchanged
                        0009   212 ;                R0      SS$_...   (may return this code as QIO status if low
                        0009   213 ;                                   bit is clear)
                        0009   214 ;
                        0009   215 ;                All other regs are preserved
                        0009   216 ;
                        0009   217 ;-
                        0009   218 CNF$PRE_QIO::                          ; QIO pre-processing for database
                        0009   219
03FE 8F   BB            0009   220          PUSHR   #^M<R1,R2,R3,R4,R5,R6,R7,R8,R9> ; Save regs
  18 BB   16            000D   221          JSB     @CNR$L_ACT_QIO(R11)             ; Setup database
03FE 8F   BA            0010   222          POPR    #^M<R1,R2,R3,R4,R5,R6,R7,R8,R9> ; Restore regs
          05            0014   223          RSB                                     ; Done
```

```
                     0015   225          .SBTTL  CNF$DELETE - Delete a CNF entry
                     0015   226 ;+
                     0015   227 ; CNF$DELETE      - Attempt to delete CNF entry
                     0015   228 ;
                     0015   229 ; The CNF is checked to see if it is delete-able.  If so, it is marked
                     0015   230 ; temporary.  If the CNF$V_FLG_ACP bit is set then the CNF does not exist in
                     0015   231 ; the linked list portion of the database and the operation is considered to
                     0015   232 ; be a no-op (these CNF's are sometimes referred to as "phantom" CNF's and
                     0015   233 ; are used to reference things known to NETACP but never inserted into the
                     0015   234 ; database: for instance, a node which was never defined but which is
                     0015   235 ; reachable by the Transport layer).
                     0015   236 ;
                     0015   237 ;
                     0015   238 ; INPUTS:        R11     CNR pointer
                     0015   239 ;                 R10     CNF pointer
                     0015   240 ;
                     0015   241 ; OUTPUTS:       R0      SS$_WRITLCK if the item was not delete-able
                     0015   242 ;                         SS$_NORMAL  otherwise
                     0015   243 ;
                     0015   244 ;                 All other regs are preserved.
                     0015   245 ;-
                     0015   246 CNF$DELETE::                                        ; Mark CNF for delete
          03BE 8F    BB  0015   247          PUSHR   #^M<R1,R2,R3,R4,R5,R7,R8,R9>    ; Save regs
       7E  0000'8F   3C  0019   248          MOVZWL  #SS$_WRITLCK,-(SP)              ; Assume not delete-able
    15 0B AA   02    E0  001E   249          BBS     #CNF$V_FLG_ACP,CNF$B_FLG(R10),30$; If BS then this is a no-op
          5B   5A    D1  0023   250          CMPL    R10,R11                         ; Is the CNF actually the CNR?
               13    13  0026   251          BEQL    50$                             ; If EQL then cannot delete
            28 BB    16  0028   252          JSB     @CNR$L_ACT_DELETE(R11)          ; Call action routine for
                     002B   253                                                     ; special processing
            0D 50    E9  002B   254          BLBC    R0,50$                          ; If LBC then cannot delete it
                     002E   255 10$:         SETBIT  CNF$V_FLG_DELETE,CNF$B_FLG(R10) ; Mark it for delete
                     0032   256              SETBIT  NET$V_PURGE,NET$GL_FLAGS        ; Remember to purge the database
       6E    00'   D0  0038   257 30$:         MOVL    S^#SS$_NORMAL,(SP)              ; Overlay status code
       03BF 8F     BA  003B   258 50$:         POPR    #^M<R0,R1,R2,R3,R4,R5,R7,R8,R9> ; Restore regs
                     05  003F   259          RSB
```

```
                     0040   261          .SBTTL  CNF$PURGE - Drain CNF entries marked for delete
                     0040   262 ;+
                     0040   263 ; CNF$PURGE       - Drain temporary entries from CNF queue
                     0040   264 ;
                     0040   265 ; The CNF is queue is scanned, starting at the root, and all CNFs which
                     0040   266 ; are marked temporary are deleted.
                     0040   267 ;
                     0040   268 ;
                     0040   269 ; INPUTS:        R11     CNR pointer
                     0040   270 ;
                     0040   271 ; OUTPUTS:       All regs are preserved.
                     0040   272 ;
                     0040   273 ;-
                     0040   274 CNF$PURGE::                                  ; Deallocate all temporary CNFs
 2C BB   16          0040   275          JSB     @CNR$L_ACT_REMOVE(R11)      ; Call action routine to do work
         05          0043   276          RSB
```

```
                        0044   278          .SBTTL  CNF$INSERT - Insert/Replace a CNF entry
                        0044   279 ;+
                        0044   280 ; CNF$INSERT    - Insert/Replace a database CNF entry
                        0044   281 ;
                        0044   282 ; Build a copy of the new CNF from the process pool and insert it into
                        0044   283 ; the database.
                        0044   284 ;
                        0044   285 ; NOTE:
                        0044   286 ;       ***  The database scan co-routine dialogue  ***
                        0044   287 ;       ***  below must be abortable via a RET.     ***
                        0044   288 ;
                        0044   289 ; INPUT:         R11     CNR pointer
                        0044   290 ;                 R10     Points to the utility buffer with new image in it
                        0044   291 ;                 R6      Pointes to old CNF entry if any
                        0044   292 ;
                        0044   293 ; OUTPUT:        R11     CNR pointer
                        0044   294 ;                 R10     Points to new CNF if successful
                        0044   295 ;                         Contains original R6 otherwise
                        0044   296 ;                 R9      Field i.d. which qualifies the error code in R0
                        0044   297 ;                 R0      Status
                        0044   298 ;
                        0044   299 ;                 All other regs contain garbage
                        0044   300 ;-
                        0044   301 CNF$INSERT::                                ; Insert/Replace a database entry
0000'CF    DD           0044   302          PUSHL   NET$GL_FLAGS               ; Save current flags
                        0048   303          SETBIT  NET$V_INTRNL,NET$GL_FLAGS ; Setup for "internal" access
                        004E   304          ;
                        004E   305          ;   Apply default values to selected parameters
                        004E   306          ;
      56    DD          004E   307          PUSHL   R6                        ; Save reg
    20 BB     16        0050   308          JSB     @CNR$L_ACT_DFLT(R11)      ; Call action routine
      56  8ED0          0053   309          POPL    R6                        ; Restore reg
    1E 50     E9        0056   310          BLBC    R0,17$                    ; If LBC then error encountered
                        0059   311          ;
                        0059   312          ;   Make sure all required fields are active
                        0059   313          ;
  52  0080 CB     9E    0059   314          MOVAB   CNR$L_VEC_MAND(R11),R2    ; Get pointer to list of field i.d.s
      59   82     D0    005E   315 10$:      MOVL    (R2)+,R9                  ; Get next field i.d.
           17     13    0061   316          BEQL    20$                       ; If EQL then done
         06CB     30    0063   317          BSBW    GET_DSC_1                 ; Get descriptor of field
   03 63   0E     E1    0066   318          BBC     #CNR$V_SEM_RT,(R3),15$    ; Br if "real" CNF field
         060B     30    006A   319          BSBW    GET_RT_FIELD              ; Else get the info from action routine
EC 18 AA   55     E0    006D   320 15$:      BBS     R5,CNF$L_MASK(R10),10$    ; If BS then field is active
 50   0000'8F     3C    0072   321          MOVZWL  #SS$_INSFARG,R0           ; Setup error status
           0070   31    0077   322 17$:      BRW     40$                       ; Take common exit
                        007A   323 20$:      ;
                        007A   324          ;   Build a list of all parameters required to be unique and scan the
                        007A   325          ;   database to see if they are in fact unique.  This list is built in
                        007A   326          ;   the CNF pointed to by R10 since this is expected to be the utility
                        007A   327          ;   buffer and should be large enough (this eliminates the need for
                        007A   328          ;   another rather large buffer).
                        007A   329          ;
    52   0C AA     3C    007A   330          MOVZWL  CNF$W_OFF_FREE(R10),R2    ; Get self-relative offset
53   0C AA42     9E    007E   331          MOVAB   CNF$W_OFF_FREE(R10)[R2],R3 ; Get ptr to free space
      55   53     D0    0083   332          MOVL    R3,R5                     ; Save copy of pointer
    52   0E AA     3C    0086   333          MOVZWL  CNF$W_SIZ_FREE(R10),R2    ; Get amount of free space
      52   04     A2    008A   334          SUBW    #4,R2                     ; Account for end of list flag
```

```
                31   19  008D   335          BLSS    32$                             ; If LSS then no space left
    54  00E4 CB   9E  008F   336          MOVAB   CNR$L_VEC_UNIQ(R11),R4  ; Get pointer to list of field i.d.s
        63   00   D0  0094   337 30$:       MOVL    #0,(R3)                         ; Mark end of list
        59   84   D0  0097   338          MOVL    (R4)+,R9                        ; Get next field i.d.
             2B   13  009A   339          BEQL    35$                             ; If EQL then at end of list
           0383   30  009C   340          BSBW    CNF$GET_FIELD                   ; Get the field value
         F2   50   E9  009F   341          BLBC    R0,30$                          ; If not active then ignore it
        52    0C   A2  00A2   342          SUBW    #12,R2                          ; Need 12 more bytes
             19   19  00A5   343          BLSS    32$                             ; If LSS the no space left
           0064   30  00A7   344          BSBW    SPCSCAN                         ; Try to do a special scan of key
    08  50   01   E0  00AA   345          BBS     #1,R0,31$                       ; Br if key recognized
        83   59   D0  00AE   346          MOVL    R9,(R3)+                        ; Else, Enter field i.d.
        83   57   7D  00B1   347          MOVQ    R7,(R3)+                        ; Enter field value/descriptor
             DE   11  00B4   348          BRB     30$                             ; Loop
                      00B6   349 31$:     ;
                      00B6   350          ;   Special lookup routine recognized the key, check status
                      00B6   351          ;
                      00B6   352          ;       R0 = Bit 0: Set if CNF found with key, else clear.
                      00B6   353          ;            Bit 1: Set if key is recogized, else clear.
                      00B6   354          ;
         DB  50   E9  00B6   355          BLBC    R0,30$                          ; Loop, if okay
    50   0000'8F   3C  00B9   356          MOVZWL  #SS$_DEVACTIVE,R0               ; Else, setup error return code
             2A   11  00BE   357          BRB     40$                             ; Take common exit
    50   0000'8F   3C  00C0   358 32$:     MOVZWL  #SS$_INSFMEM,R0                 ; Setup status code
             23   11  00C5   359          BRB     40$                             ; Take common exit
                      00C7   360          ;
            00000004  00C7   361 35$:     DLIST = 4                               ; Offset for dynamic field lis pointer
            00000008  00C7   362          SLIST = 8                               ; Offset for static field list pointer
                      00C7   363          PUSHQ   R4                              ; Dynamic pointer is garbage,
                      00CA   364          ;                                       ; Static pointer is in R5
  29'AF     02   FB  00CA   365          CALLS   #2,B^SCAN                       ; Scan for field already in use
        19  50   E9  00CE   366          BLBC    R0,40$                          ; If LBC then something's not unique
                      00D1   367          ;
                      00D1   368          ;  Create a copy of the new CNF
                      00D1   369          ;
           00CC   30  00D1   370          BSBW    CNF$CLONE                       ; Create a copy - clone returns in R10
         13  50   E9  00D4   371          BLBC    R0,40$                          ; If LBC then error
       0C40  8F   BB  00D7   372          PUSHR   #^M<R6,R10,R11>                 ; Save critical regs
         24  BB   16  00DB   373          JSB     @CNR$L_ACT_INSERT(R11)  ; Perform any pre-insertion processing
       0C40  8F   BA  00DE   374          POPR    #^M<R6,R10,R11>                 ; Restore regs
         0A  50   E8  00E2   375          BLBS    R0,45$                          ; If LBS then successful
0000'DF     6A   0E  00E5   376          INSQUE  (R10),@NET$GQ_TMP_BUF   ; Else queue "new" CNF for deallocation
                      00EA   377 40$:     ;
                      00EA   378          ;   Since the insert operation has failed, copy the old CNF pointer to
                      00EA   379          ;   R10 since R10 is used to return the CNF representing this entry
                      00EA   380          ;   which is linked into the database regardless of the success or
                      00EA   381          ;   failure of the attmepted insertion.  R10 will return the value
                      00EA   382          ;   zero if there was no old CNF pointer.
                      00EA   383          ;
    5A    56   D0  00EA   384          MOVL    R6,R10                          ; Copy the "old" CNF pointer
             0B   11  00ED   385          BRB     70$                             ; Take common exit
                      00EF   386 45$:     ;
                      00EF   387          ;   Insert the new CNF into the database
                      00EF   388          ;
       0C40  8F   BB  00EF   389          PUSHR   #^M<R6,R10,R11>                 ; Save critical regs
         34  BB   16  00F3   390          JSB     @CNR$L_INSERT(R11)              ; Perform the insertion
       0C40  8F   BA  00F6   391          POPR    #^M<R6,R10,R11>                 ; Restore regs
```

```
                0000'CF 8ED0   00FA    392 70$:    POPL    NET$GL_FLAGS            ; Restore flags
                   0B 50  E9   00FF    393         BLBC    R0,80$                  ; If LBC then error
                      01  E1   0102    394         BBC     #CNF$V_FLG_DELETE,-     ; If BC then no need to delete new
                06 0B AA        0104    395                 CNF$B_FLG(R10),80$      ; entry
                                0107    396         SETBIT  NET$V_PURGE,-           ; Else remember to purge it from the
                                0107    397                 NET$GL_FLAGS            ; database
                         05     010D    398 80$:    RSB                             ; Done
                                010E    399
                                010E    400
                                010E    401
                                010E    402
                                010E    403 SPCSCAN:                                ; Try to do special scan of database
                                010E    404             ;
                                010E    405             ;   The special lookup routine will be called to try to do a
                                010E    406             ;   "quick" lookup of the CNF, given the current key. If the
                                010E    407             ;   key is not recognized then bit 1 of R0 is returned clear.
                                010E    408             ;   If the CNF is found, then the low bit of R0 is set, else
                                010E    409             ;   it is clear.
                                010E    410             ;
                                010E    411             ;   If the key is not recognized, then the key is inserted into
                                010E    412             ;   the key list for the long scan routine to check.
                                010E    413             ;
                      5A   DD   010E    414         PUSHL   R10                     ; Save regs
                      5A   D4   0110    415         CLRL    R10                     ; Start from beginning
                   38 BB   16   0112    416         JSB     @CNR$L_SPCSCAN(R11)     ; Check for quick lookup of key
                0C 50   01   E1 0115    417         BBC     #1,R0,40$               ; Br if key not recognized
                                0119    418             ;
                                0119    419             ;   Special lookup routine recognized the key, check status
                                0119    420             ;
                                0119    421             ;       R0 = Bit 0: Set if CNF found with key, else clear.
                                0119    422             ;            Bit 1: Set if key is recogized, else clear.
                                0119    423             ;
                09 50   E9     0119    424         BLBC    R0,40$                  ; Br if not found, okay
                56 5A   D1      011C    425         CMPL    R10,R6                  ; Else, is this the same CNF?
                   04   12     011F    426         BNEQ    40$                     ; Br if no, bad CNF
                                0121    427         CLRBIT  #0,R0                   ; Else, indicate okay
                      5A 8ED0   0125    428 40$:    POPL    R10                     ; Restore regs
                         05     0128    429         RSB                             ; Take common exit
                                0129    430
                                0129    431
                                0129    432
                                0129    433
                                0129    434 ;
                                0129    435 ;  Make sure those fields whose value should be unique are unique
                                0129    436 ;
                         0400   0129    437 $CAN:   .WORD   ^M<R10>                 ;
                                012B    438 ;
                                012B    439 ;   Check if argument list is empty
                                012B    440 ;
                      50   D4   012B    441         CLRL    R0                      ; Assume success, low bit flipped below
                   04 BC   D5   012D    442         TSTL    @DLIST(AP)              ; Empty argument list?
                      34   13   0130    443         BEQL    105$                    ; Br if yes, return immediately
                                0132    444
                52    00   D0   0132    445         MOVL    #NFB$C_OP_EQL,R2        ; Get action routine index
                5A    5B   D0   0135    446         MOVL    R11,R10                 ; Start at begining of list
                   30 BB   16   0138    447         JSB     @CNR$L_SCANNER(R11)     ; Call scanner to prepare scan
                                013B    448 60$:    ;
```

```
                        013B   449          ;   Get next CNF block
                        013B   450          ;
          50   00   D0  013B   451          MOVL    #CNF$_ADVANCE,R0        ; Say "Give me the next CNF"
               9E   16  013E   452          JSB     @(SP)+                  ; Tell co-routine, he calls us back
                        0140   453                                          ; with a JSB @(SP)+ and status in R0
            23 50   E9  0140   454          BLBC    R0,100$                 ; If LBC there was none
          56   5A   D1  0143   455          CMPL    R10,R6                  ; Is this the CNF being replaced?
               F3   13  0146   456          BEQL    60$                     ; If EQL yes, ignore it
    04 AC    08 AC   D0  0148   457          MOVL    SLIST(AP),DLIST(AP)     ; Start at the top of parameter list
                        014D   458 70$:      ;
                        014D   459          ;   See if any fields in the list match the any of the fields in the
                        014D   460          ;   CNF already in the database.
                        014D   461          ;
          50   04 AC   D0  014D   462          MOVL    DLIST(AP),R0            ; Get pointer to next parameter
          59   80   D0  0151   463          MOVL    (R0)+,R9                ; Get parameter i.d.
               E5   13  0154   464          BEQL    60$                     ; If EQL then done with this CNF block
          57   80   7D  0156   465          MOVQ    (R0)+,R7                ; Get parameter value/descriptor
    04 AC      50   D0  0159   466          MOVL    R0,DLIST(AP)            ; Store pointer
             05D1   30  015D   467          BSBW    GET_DSC_1               ; Get field semantics
             01E4   30  0160   468          BSBW    COMPARE                 ; Make field comparison
            E7 50   E9  0163   469          BLBC    R0,70$                  ; If no match, loop on next field
                        0166   470          ;
                        0166   471 100$:     ;
                        0166   472          ;   We are done.  The RET instruction aborts the scanner co-routine.
                        0166   473          ;
       05 50    00   E3  0166   474 105$:    BBCS    #0,R0,110$              ; If BC in R0 then no unique field
                        016A   475                                          ; violations were detected
    50   0000'8F   3C  016A   476          MOVZWL  #SS$_DEVACTIVE,R0       ; Indicate unique field violation
                   04   016F   477 110$:    RET                             ; Return status in R0
```

```
                                                E 5
                     0170   479          .SBTTL  CNF$COPY - Copy a CNF to another
                     0170   480 ;+
                     0170   481 ; CNF$COPY        - Copy one CNF entry into another
                     0170   482 ;
                     0170   483 ; The contents of a source CNF block are copied to the destination CNF block.
                     0170   484 ; No string storage compression takes place, but any additional storage space
                     0170   485 ; in the destination CNF block are reflected in its CNF$W_SIZ_FREE field.
                     0170   486 ;
                     0170   487 ; INPUTS:        R11     CNR pointer
                     0170   488 ;                 R10     Destination CNF pointer
                     0170   489 ;                 R8      Source CNF pointer
                     0170   490 ;
                     0170   491 ; OUTPUTS:       R0      SS$_NORMAL  if successful
                     0170   492 ;                         SS$_INSFMEM if destination CNF is too small
                     0170   493 ;
                     0170   494 ;                 All other registers are preserved.
                     0170   495 ;-
                     0170   496 CNF$COPY::
        007E 8F   BB 0170   497          PUSHR   #^M<R1,R2,R3,R4,R5,R6>      ; Save regs
    50  0000'8F   3C 0174   498          MOVZWL  #SS$_INSFMEM,R0             ; Assume destination CNF is too small
    56    08 AA   3C 0179   499          MOVZWL  CNF$W_SIZE(R10),R6          ; Save size of target CNF
    08 A8   56   B1 017D   500          CMPW    R6,CNF$W_SIZE(R8)           ; Is it big enough?
            18   1F 0181   501          BLSSU   10$                         ; If LSS then too small
6A  68    08 A8   28 0183   502          MOVC3   CNF$W_SIZE(R8),(R8),(R10)   ; Copy CNF
    08 AA   56   B0 0188   503          MOVW    R6,CNF$W_SIZE(R10)          ; Restore original size
    56    08 A8   A2 018C   504          SUBW    CNF$W_SIZE(R8),R6           ; Get difference in size
    0E AA   56   A0 0190   505          ADDW    R6,CNF$W_SIZ_FREE(R10)      ; Update the amount of free space
            8A   94 0194   506          BICB    #CNF$M_FLG_CNR!-            ; Block is not a CNR
                 0195   507                   CNF$M_FLG_DELETE!-           ; Block is a temporary CNF or marked for d
                 0195   508                   CNF$M_FLG_ACP,-              ; Block is a catch-all used by the ACP
    08 AA   07        0195   509                   CNF$B_FLG(R10)               ; Init flags
    50   00'   D0 0198   510          MOVL    S^#SS$_NORMAL,R0            ; Indicate success
    007E 8F   BA 019B   511 10$:     POPR    #^M<R1,R2,R3,R4,R5,R6>      ; Restore regs
              05 019F   512          RSB                                 ; Done
```

```
                                01A0    514          .SBTTL  CNF$CLONE - Compress a CNF entry
                                01A0    515 ;+
                                01A0    516 ; CNF$CLONE       - Create a compressed version of a CNF entry
                                01A0    517 ;
                                01A0    518 ; A resultant CNF block is allocated and initialized.  The contents of a source
                                01A0    519 ; CNF block are copied to it such that the string storage space is
                                01A0    520 ; unfragmented.
                                01A0    521 ;
                                01A0    522 ; INPUTS:        R11     CNR pointer
                                01A0    523 ;                 R10     Source CNF pointer -- usually utility buffer
                                01A0    524 ;
                                01A0    525 ; OUTPUTS:       R10     New CNF address -- the old R10 value is lost
                                01A0    526 ;                 R0      SS$_NORMAL  if successful
                                01A0    527 ;                         SS$_INSFMEM otherwise
                                01A0    528 ;
                                01A0    529 ;                 All other registers are preserved.
                                01A0    530 ;-
                                01A0    531 CNF$CLONE::                                 ; Create a compressed copy of a CNF
                     007E 8F    BB   01A0    532          PUSHR   #^M<R1,R2,R3,R4,R5,R6>  ; Save regs
                     56   5A    D0   01A4    533          MOVL    R10,R6                  ; Save a pointer to the old CNF
                                01A7    534          ;
                                01A7    535          ;   Allocate new CNF block and initialize its fixed portion
                                01A7    536          ;
                50   0000'8F    3C   01A7    537          MOVZWL  #SS$_INSFMEM,R0         ; Assume destination CNF is too small
                          5A    D4   01AC    538          CLRL    R10                     ; Zero pointer to the new CNF
                  51    0C AB   3C   01AE    539          MOVZWL  CNR$W_SIZ_CNF(R11),R1   ; Get minimum block size
                  51    10 A6   A0   01B2    540          ADDW    CNF$W_SIZ_USED(R6),R1   ; Add in string space used
                             23   1D   01B6    541          BVS     10$                     ; If VS the >65K
                           FE45'  30   01B8    542          BSBW    NET$ALLOCATE            ; Allocate block from ACP pool
                        5A 50     E9   01BB    543          BLBC    R0,100$                 ; Br on error
                     5A    52     D0   01BE    544          MOVL    R2,R10                  ; Copy block pointer
                           51     DD   01C1    545          PUSHL   R1                      ; Save size
                           0C AB  2C   01C3    546          MOVC5   CNR$W_SIZ_CNF(R11),-    ; Copy the fixed portion of the block
                62   51     00    66   01C6    547                  (R6),#0,RT,(R2)        ; and zero the remainder
                        08 AA     8E   F7   01CA    548          CVTLW   (SP)+,CNF$W_SIZE(R10)   ; Store size for deallocation
                                       8A   01CE    549          BICB    #CNF$M_FLG_CNR!-        ; Block is not a CNR
                                               01CF    550                   CNF$M_FLG_DELETE!-     ; Block is a temporary CNF or marked for del
                                               01CF    551                   CNF$M_FLG_ACP,-        ; Block is a catch-all used by the ACP
                        0B AA     07         01CF    552                   CNF$B_FLG(R10)         ; Init flags
                              005F   30   01D2    553          BSBW    CNF$INIT                ; Init remainder of CNF
                        55    0E AB  3C   01D5    554          MOVZWL  CNR$W_MAX_INX(R11),R5   ; Get max field index
                              37     11   01D9    555          BRB     40$                     ; Jump to the end of the loop
                                       01DB    556 10$:       ;
                                       01DB    557            ;   Find the next string field
                                       01DB    558            ;
             53    0128 CB45     DE   01DB    559          MOVAL   CNR$L_SEM_TAB(R11)[R5],R3 ; Get address of field semantics
                           08      ED   01E1    560          CMPZV   #CNR$V_SEM_TYP,-        ; Is it for strings ?
                     63    03         01E3    561                  #CNR$S_SEM_TYP,(R3),-  ;
                           04         01E5    562                  #CNR$C_SEM_STR         ;
                           2A     12   01E6    563          BNEQ    40$                     ; If not branch to try next field
                                       01E8    564          ;
                                       01E8    565          ;   Move the string if its active.  Clear the mask bit before the call
                                       01E8    566          ;   to PUT_STR so that the CNF$W_SIZ_USED is not erroneously updated.
                                       01E8    567          ;
          25 18 AA     55     E5   01E8    568          BBCC    R5,CNF$L_MASK(R10),40$  ; Br if field is not active
                21 63     0E     E0   01ED    569          BBS     #CNR$V_SEM_RT,(R3),40$  ; Br if "field" is actually a routine
                           00     EF   01F1    570          EXTZV   #CNR$V_SEM_OFF,-        ; Get byte offset from top of
```

```
                                01F3   571                     #CNR$S_SEM_OFF,(R3),R1  ; CNF to the field
50   51   56   C1   01F6   572          ADDL3   R6,R1,R0        ; Get source CNF field address
     51   5A   C0   01FA   573          ADDL    R10,R1          ; Get dest.  CNF field address
                     01FD   574          ;
                     01FD   575          ;   Move the string to the new CNF
                     01FD   576          ;
     58   60   3C   01FD   577          MOVZWL  STR_OFF(R0),R8  ; Get self-relative offset to string
     58   50   C0   0200   578          ADDL    R0,R8           ; Make it a pointer
  57   02 A0   3C   0203   579          MOVZWL  STR_LNG(R0),R7  ; Get its size
       03B3    30   0207   580          BSBW    PUT_STR         ; Store it
     0B 50    E9   020A   581          BLBC    R0,TOO$         ; If LBC then error
00 18 AA   55   E2   020D   582          BBSS    R5,CNF$L_MASK(R10),40$ ; Mark the field valid
     C6 55    F4   0212   583 40$:     SOBGEQ  R5,10$          ; Loop for each field
                     0215   584          ;
                     0215   585          ;   Done
                     0215   586          ;
     50   00'  3C   0215   587          MOVZWL  S^#SS$_NORMAL,R0  ; Indicate success
     007E 8F   BA   0218   588 100$:    POPR    #^M<R1,R2,R3,R4,R5,R6>  ; Restore regs
                05   021C   589          RSB
```

```
                                                H 5
                     021D   591          .SBTTL  CNF$INIT - Initialize CNF entry
                     021D   592 ;+
                     021D   593 ; CNF$INIT         - Initialize CNF entry
                     021D   594 ; CNF$INIT_UTL     - Initialize the utility buffer as a CNF entry
                     021D   595 ;
                     021D   596 ; A CNF block is initialized.
                     021D   597 ;
                     021D   598 ; INPUTS:        R11     CNR pointer
                     021D   599 ;                R10     If CNF$INIT then ptr to CNF block to be initialized.
                     021D   600 ;                        If CNF$INIT_UTL then scratch
                     021D   601 ;
                     021D   602 ; OUTPUTS:       R10     If CNF$INIT then unchanged.
                     021D   603 ;                        If CNF$INIT_UTL then ptr to utility buffer
                     021D   604 ;                R0      SS$_NORMAL  if successful
                     021D   605 ;                        SS$_INSFMEM if CNF block is too small
                     021D   606 ;
                     021D   607 ;                All other registers are preserved.
                     021D   608 ;-
                     021D   609 CNF$INIT_UTL::                           ; Init utility buffer as a CNF BLOCK
5A  0000'CF   D0     021D   610          MOVL    NET$GL_UTLBUF,R10       ; Point to the utility buffer
      1000 8F   B0   0222   611          MOVW    #NET$C_UTLBUFSIZ,-      ; Setup its size
        08 AA        0226   612                      CNF$W_SIZE(R10)
                     0228   613
                     0228   614          ASSUME  CNR$C_MAX_INX  EQ  95   ; One bit in mask for each parameter
                     0228   615                                          ; index (95 (zero indexed) => 3 lwords)
        18 AA   7C   0228   616          CLRQ    CNF$L_MASK(R10)         ; Clear first 2 mask longwords
        20 AA   D4   022B   617          CLRL    CNF$L_MASK+8(R10)       ; Clear third mask longword
        12 AA   B4   022E   618          CLRW    CNF$W_ID(R10)           ; Init CNF i.d. data
        0B AA   94   0231   619          CLRB    CNF$B_FLG(R10)          ; Zero all flags
                     0234   620
                     0234   621
                     0234   622 CNF$INIT::                               ; Initialize a CNF block
50  0000'8F   3C     0234   623          MOVZWL  #SS$_INSFMEM,R0         ; Assume error
        0C AB   B1   0239   624          CMPW    CNR$W_SIZ_CNF(R11),-    ; Is block big enough ?
        08 AA        023C   625                      CNF$W_SIZE(R10)
             17   1A 023E   626          BGTRU   10$                     ; If GTRU then CNF is too small
             17   90 0240   627          MOVB    #DYN$C_NET,-
        0A AA        0242   628                      CNF$B_TYPE(R10)     ; Enter type
        10 AA   B4   0244   629          CLRW    CNF$W_SIZ_USED(R10)     ; Init free spaced used for strings
             0C   A3 0247   630          SUBW3   #CNF$W_OFF_FREE,-       ; Setup self-relative offset to free
        0C AB        0249   631                      CNR$W_SIZ_CNF(R11),-; space
        0C AA        024B   632                      CNF$W_OFF_FREE(R10)
        0C AB   A3   024D   633          SUBW3   CNR$W_SIZ_CNF(R11),-    ; Setup amount of free space available
        08 AA        0250   634                      CNF$W_SIZE(R10),-
        0E AA        0252   635                      CNF$W_SIZ_FREE(R10)
     50   00'  D0   0254   636          MOVL    S^#SS$_NORMAL,R0        ; Indicate success
                05   0257   637 10$:    RSB
```

```
                                                                I  5
                     0258    639          .SBTTL  CNF$KEY_SEARCH - Search for selected CNFs
                     0258    640 ;+
                     0258    641 ; CNF$KEY_SRCH_EX  - External find CNF via match of supplied parameter
                     0258    642 ; CNF$KEY_SEARCH   - Internal find CNF via match of supplied parameter
                     0258    643 ;
                     0258    644 ; The CNF list is search until a block is found in which the supplied key
                     0258    645 ; matches the appropriate field.  A match is determined by dispatching to the
                     0258    646 ; compare routine identified by R1.
                     0258    647 ;
                     0258    648 ; If R10 is zero on input then the search begins at the CNR (root), else R10
                     0258    649 ; is assumed to be the address of a CNF and the search begins with the CNF
                     0258    650 ; following the R10 CNF.
                     0258    651 ;
                     0258    652 ; INPUTS:        R11 = CNR address
                     0258    653 ;                R10 = CNF address or zero
                     0258    654 ;                R9  = FLD # in bits 0-15, Mask ID in bits 16-23
                     0258    655 ;                      (or NFB$C_WILDCARD to match any CNF entry)
                     0258    656 ;                R8  = Key value if bit, byte, word, or longword parameter type
                     0258    657 ;                      Key pointer if key is a string
                     0258    658 ;                R7  = Key length if key is a string
                     0258    659 ;                R1  = Search function
                     0258    660 ;                R0  = Error code to be returned if CNF is not found
                     0258    661 ;
                     0258    662 ;                R7/R8 are not supplied if R1 = NFB$C_OP_FNDMIN or FNDMAX.
                     0258    663 ;
                     0258    664 ; OUTPUTS:       R10 = Address of matching CNF if search is successful, else 0
                     0258    665 ;                R1  = Garbage
                     0258    666 ;                R0  = Low bit set if search is successful
                     0258    667 ;                      Unchanged otherwise (SS$_ENDOFFILE if entered with LBS)
                     0258    668 ;
                     0258    669 ;                All other registers are preserved
                     0258    670 ;
                     0258    671 ;-
                     0258    672 CNF$KEY_SRCH_EX::                           ; Locate CNF via key
 7E  D4              0258    673          CLRL    -(SP)                      ; Terminate key list
 7E  7D              025A    674          MOVQ    R7,-(SP)                   ; Store key value
 51  DD              025D    675          PUSHL   R1                         ; Store type of comparison
 59  DD              025F    676          PUSHL   R9                         ; Store field ID
 51  5E  D0          0261    677          MOVL    SP,R1                      ; Set address of key list
     16  10          0264    678          BSBB    CNF$SEARCH_EX              ; Call external search routine
 5E  14  C0          0266    679          ADDL    #5*4,SP                    ; Cleanup key list
         05          0269    680          RSB
                     026A    681
                     026A    682 CNF$KEY_SEARCH::                           ; Locate CNF via key
 7E  D4              026A    683          CLRL    -(SP)                      ; Terminate key list
 7E  7D              026C    684          MOVQ    R7,-(SP)                   ; Store key value
 51  DD              026F    685          PUSHL   R1                         ; Store type of comparison
 59  DD              0271    686          PUSHL   R9                         ; Store field ID
 51  5E  D0          0273    687          MOVL    SP,R1                      ; Set address of key list
     10  10          0276    688          BSBB    CNF$SEARCH                 ; Call internal search routine
 5E  14  C0          0278    689          ADDL    #5*4,SP                    ; Cleanup key list
         05          027B    690          RSB
```

```
                        .SBTTL  CNF$SEARCH - Search for CNFs by list of keys
;+
; CNF$SEARCH_EX  - External find CNF via match of supplied list of keys
; CNF$SEARCH    - Internal find CNF via match of supplied list of keys
;
; The CNF list is searched until a block is found in which the supplied list
; of search keys matches the appropriate fields.  The list of keys supplies
; the field IDs to be compared, the type of comparision for each field, and
; the actual key value.  The CNF is matched if all of the search keys match
; the appropriate fields in the CNF (AND-type search).
;
; If R10 is zero on input then the search starts at the beginning.  Else R10
; is assumed to be the address of a CNF and the search begins with the CNF
; following the R10 CNF.
;
;
; To optimize the search of a database, if there is only one key and the
; operator is EQL then we will call a special SCAN routine to try to optimize
; lookups.
;
;
; Inputs:
;
;       R11 = CNR address
;       R10 = Starting CNF address, or zero
;       R0  = Error code to be returned if CNF is not found
;       R1  = Address of a list of search keys:
;
;                          +----------------------------+
;                          |     First field ID         |
;                          +----------------------------+
;                          |   Type of comparison       |     (NFB$C_OP_xxx)
;                          +----------------------------+
;                          |     Search key value       |     (8 bytes)
;                          | (descriptor or longword)   |
;                          +============================+
;                          |     Second field ID        |
;                          +----------------------------+
;                          |   Type of comparison       |
;                          +----------------------------+
;                          |     Secondary key value    |
;                          | (descriptor or longword)   |
;                          +============================+
;                          |            .               |     (repeat for each key)
;                          |            .               |
;                          |            .               |
;                          +============================+
;                          |            0               |     (terminates list)
;                          +----------------------------+
;
;       If the FNDMIN, FNDMAX or FNDPOS operators are used, then only
;       one search key is allowed.
;
;       The key value quadword in the key list is ignored when used with
;       the FNDMIN or FNDMAX operators.
;
; Outputs:
```

```
                                027C   749 ;
                                027C   750 ;       R11 = Address of CNR
                                027C   751 ;       R10 = Address of matching CNF if search is successful, else 0
                                027C   752 ;       R0  = Low bit set if search is successful
                                027C   753 ;             Unchanged otherwise (SS$_ENDOFFILE if entered with LBS)
                                027C   754 ;
                                027C   755 ;       All registers are preserved.
                                027C   756 ;-
                                027C   757 CNF$SEARCH_EX::                           ; Locate CNF via list of keys
                0000'CF   DD    027C   758         PUSHL   NET$GL_FLAGS             ; Save current flags
                          0280  759         CLRBIT  NET$V_INTRNL,NET$GL_FLAGS ; Indicate external access rights
                      0A  11    0286   760         BRB     SEARCH
                                0288   761
                                0288   762 CNF$SEARCH::                            ; Locate CNF via list of keys
                0000'CF   DD    0288   763         PUSHL   NET$GL_FLAGS             ; Save current flags
                          028C  764         SETBIT  NET$V_INTRNL,NET$GL_FLAGS ; Indicate internal access rights
                                0292   765
                                0292   766 SEARCH:
                                0292   767         SETBIT  NET$V_READ,NET$GL_FLAGS ; Access will be for read only
                   05 50   E9   0298   768         BLBC    R0,10$                   ; Invalid error code if LBS
           50   0000'8F   3C   029B   769         MOVZWL  #SS$_ENDOFFILE,R0        ; Make it a valid error code
                03FF 8F   BB   02A0   770 10$:    PUSHR   #^M<R0,R1,R2,R3,R4,R5,R6,R7,R8,R9> ; Save regs and default error sta
                               02A4   771         ;
                               02A4   772         ;   If there is only one key, and that operator is EQL then
                               02A4   773         ;   we will call the special scan routine. OR if there are two
                               02A4   774         ;   search keys and the second is a WILDCARD.
                               02A4   775         ;
              04 A1    00      D1   02A4   776         CMPL    #NFB$C_OP_EQL,4(R1)      ; Is this an equals operation?
                       28      12   02A8   777         BNEQ    15$                      ; Br if not, general scan
                    10 A1      D5   02AA   778         TSTL    16(R1)                   ; Only one search key?
                       0B      13   02AD   779         BEQL    13$                      ; Br if yes, do special lookup
              10 A1    01      D1   02AF   780         CMPL    #NFB$C_WILDCARD,16(R1)   ; Is the second a wildcard?
                       1D      12   02B3   781         BNEQ    15$                      ; Br if not
                    20 A1      D5   02B5   782         TSTL    32(R1)                   ; Is this the end?
                       18      12   02B8   783         BNEQ    15$                      ; Br if not, do complete lookup
                 59    61      D0   02BA   784 13$:    MOVL    (R1),R9                  ; Get the search field ID
              57    08 A1      7D   02BD   785         MOVQ    8(R1),R7                 ; Get the search key value/desc.
                       51      DD   02C1   786         PUSHL   R1                       ; Save address of key list
                    38 BB      16   02C3   787         JSB     @CNR$L_SPCSCAN(R11)      ; Else, do special scan
                       51   8ED0   02C6   788         POPL    R1                       ; Restore address of key list
              05 50    01      E1   02C9   789         BBC     #1,R0,15$                ; Br if the key not recognized
                    6A 50      E8   02CD   790         BLBS    R0,79$                   ; Br on success, else fall thru
                       6B      11   02D0   791         BRB     80$                      ; Else, return error
                 56    51      D0   02D2   792 15$:    MOVL    R1,R6                    ; Copy address of key list
                               02D5   793         ;
                               02D5   794         ;   Call co-routine to prepare for scan
                               02D5   795         ;
                    30 BB      16   02D5   796         JSB     @CNR$L_SCANNER(R11)      ; Initialize scanner co-routine
                               02D8   797         ;
                               02D8   798         ;   Initialize min/max selection storage (OP_FNDMIN or OP_FNDMAX only)
                               02D8   799         ;
                0000'CF    D4   02D8   800         CLRL    SELECT_CNF               ; Indicate no CNF matched
                0004'CF    D4   02DC   801         CLRL    SELECT_VALUE             ; Make current min/max a null string
           0008'CF    01   CE   02E0   802         MNEGL   #1,SELECT_VALUE+4        ; Make current min/max infinity
                               02E5   803         ;
                               02E5   804         ;   Skip to the next CNF
                               02E5   805         ;
```

```
                50    00    9A  02E5    806 20$:    MOVZBL  #CNF$_ADVANCE,R0        ; Say "Give me the next CNF"
                    9E    16  02E8    807         JSB     @(SP)+                  ; Tell co-routine, he calls us back
                                02EA    808                                         ; with a JSB @(SP)+ and status in R0
                  28 50   E9  02EA    809         BLBC    R0,70$                  ; If LBC there was none
                                02ED    810         ;
                                02ED    811         ;  Using the list of keys, compare each of the key values with the
                                02ED    812         ;  corresponding fields in the CNF to determine if the CNF matches.
                                02ED    813         ;
                52    56    D0  02ED    814         MOVL    R6,R2                   ; Pick up original keylist pointer
                59    82    D0  02F0    815 25$:    MOVL    (R2)+,R9                ; Get next search field ID
                      37    13  02F3    816         BEQL    60$                     ; If none left, then we matched!
                      82    D5  02F5    817         TSTL    (R2)+                   ; Skip type of comparison for now
                57    82    7D  02F7    818         MOVQ    (R2)+,R7                ; Get search key value
                01    59    D1  02FA    819         CMPL    R9,#NFB$C_WILDCARD      ; Wildcard search key?
                      F1    13  02FD    820         BEQL    25$                     ; If so, then match this field
                   03D7     30  02FF    821         BSBW    GET_DSC                 ; On return:
                10 50       E9  0302    822         BLBC    R0,70$                  ;  R10 = addr of CNF ptr
                                0305    823                                         ;  R5  = bit offset to bit from the
                                0305    824                                         ;       top of mask vector
                                0305    825                                         ;  R4  = offset to parameter from top
                                0305    826                                         ;       of CNF, or routine address
                                0305    827                                         ;  R3  = ptr to field semantics
                                0305    828                                         ;  R0  = LBS if successful
                      52    DD  0305    829         PUSHL   R2                      ; Save pointer into key list
            52    F4 A2       D0  0307    830         MOVL    -12(R2),R2              ; Get type of comparison for this key
                      3A    10  030B    831         BSBB    COMPARE                 ; Make field comparison
                      52 8ED0   030D    832         POPL    R2                      ; Restore key list pointer
                   D2 50    E9  0310    833         BLBC    R0,20$                  ; If key doesn't match, skip this CNF
                      DB    11  0313    834         BRB     25$                     ; If it does match, compare next field
                                0315    835         ;
                                0315    836         ;  We could not match any CNFs.  Return default error to caller.
                                0315    837         ;
                                0315    838 70$:    $DISPATCH 4(R6),<-              ; Are we searching for min/max CNF?
                                0315    839             <NFB$C_OP_FNDMIN, 75$>-     ; Branch if so
                                0315    840             <NFB$C_OP_FNDMAX, 75$>>
                50    02    9A  031E    841 72$:    MOVZBL  #CNF$_QUIT,R0           ; Say "I quit without finding CNF"
                      9E    16  0321    842         JSB     @(SP)+                  ; Tell co-routine, returns clean stack
                      18    11  0323    843         BRB     80$                     ; Exit
                                0325    844         ;
                                0325    845         ;  We have completed a full scan of the database for the operator
                                0325    846         ;  functions NFB$C_OP_FNDMIN or NFB$C_OP_FNDMAX.  Now return the
                                0325    847         ;  CNF which was determined to have the minimum or maximum value.
                                0325    848         ;
            5A  0000'CF       D0  0325    849 75$:    MOVL    SELECT_CNF,R10          ; Return selected CNF
                      F2    13  032A    850         BEQL    72$                     ; If none, return failure
                                032C    851         ;
                                032C    852         ;  We have matched a CNF.  Return it to the caller.
                                032C    853         ;
                50    03    D0  032C    854 60$:    MOVL    #CNF$_TAKE_CURR,R0      ; Say "I want this one"
                06    04 A6     D1  032F    855         CMPL    4(R6),#NFB$C_OP_FNDPOS  ; Are we searching for position?
                      03    12  0333    856         BNEQ    65$                     ; If NEQ then no
                50    01    D0  0335    857         MOVL    S^#CNF$_TAKE_PREV,R0    ; Say "I want the previous block"
                      9E    16  0338    858 65$:    JSB     @(SP)+                  ; Tell co-routine, returns clean stack
                6E    00'   D0  033A    859 79$:    MOVL    S^#SS$_NORMAL,(SP)      ; Setup success status code
                03FF  8F    BA  033D    860 80$:    POPR    #^M<R0,R1,R2,R3,R4,R5,R6,R7,R8,R9> ; Restore regs
                0000'CF 8ED0  0341    861         POPL    NET$GL_FLAGS            ; Restore flags
                          05  0346    862         RSB
```

```
                                        .SBTTL  COMPARE - Compare CNF against keys
                                ;+
                                ; COMPARE - Compare CNF against a key value
                                ;
                                ; Inputs:
                                ;
                                ;       R10 = Address of CNF
                                ;       R7/R8 = Key value
                                ;       R5  = Bit offset to "valid" bit from the top of mask vector
                                ;       R4  = Offset into CNF for parameter data
                                ;       R3  = Pointer to field semantics
                                ;       R2  = Type of comparison
                                ;
                                ; Outputs:
                                ;
                                ;       R0  = True if matched, else false.
                                ;-
                                
                                COMPARE:
                                        ;
                                        ;   The 'BSBB COMPARE_ACT"  cannot be called to setup the condition
                                        ;   codes prior to the dispatch since the $DISPATCH macro expansion
                                        ;   includes a CASE instruction which modifies the condition codes.
                                        ;
                                
                                        $DISPATCH  R2,<-
                                
                                            <NFB$C_OP_EQL,    KEY_EQL>  -; Match if EQL
                                            <NFB$C_OP_NEQ,    KEY_NEQ>  -; Match if KEY NEQ  CNF field
                                            <NFB$C_OP_GTRU,   KEY_GTRU> -; Match if KEY GTRU CNF field
                                            <NFB$C_OP_LSSU,   KEY_LSSU> -; Match if KEY LSSU CNF field
                                            <NFB$C_OP_FNDMIN, KEY_MIN>  -; Find the minimum KEY value
                                            <NFB$C_OP_FNDMAX, KEY_MAX>  -; Find the maximum KEY value
                                            <NFB$C_OP_FNDPOS, KEY_LSSU> -; Match if KEY LSSU CNF field
                                        >
                                        BUG_CHECK    NETNOSTATE,FATAL   ; Index is unknown
                                
                 3B    10       901 KEY_EQL:        BSBB    COMPARE_ACT       ; Compare the fields
                 35    13       902                 BEQL    MATCH             ; Br if KEY is EQL CNF field
                 30    11       903                 BRB     NO_MA
                                904
                 35    10       905 KEY_NEQ:        BSBB    COMPARE_ACT       ; Compare the fields
                 2F    12       906                 BNEQ    MATCH             ; Br if KEY is EQL CNF field
                 2A    11       907                 BRB     NO_MA
                                908
                 2F    10       909 KEY_GTRU:       BSBB    COMPARE_ACT       ; Compare the fields
                 29    1A       910                 BGTRU   MATCH             ; Br if KEY is GTRU CNF field
                 24    11       911                 BRB     NO_MA
                                912
                 29    10       913 KEY_LSSU:       BSBB    COMPARE_ACT       ; Compare the fields
                 23    1F       914                 BLSSU   MATCH             ; Br if KEY is LSSU CNF field
                 1E    11       915                 BRB     NO_MA
                                916
     0004'CF     7D       917 KEY_MAX:        MOVQ    SELECT_VALUE,R7 ; Get the current min/max value
                 1E    10       918                 BSBB    COMPARE_ACT       ; Compare the fields
                 15    1E       919                 BGEQU   NO_MA             ; If GEQU current KEY is still maximum
                 09    11       920                 BRB     UPD               ; Else update to new max value
```

```
                        0380   921
                        0380   922 KEY_MIN:        MOVQ    SELECT_VALUE,R7 ; Get the current min/max value
                  13    0385   923                 BSBB    COMPARE_ACT     ; Compare the fields
                  0A    0387   924                 BLEQU   NO_MA           ; If LEQU current KEY is still minimum
                        0389   925
0000'CF   5A      D0    0389   926 UPD:            MOVL    R10,SELECT_CNF  ; Update the current matched CNF
0004'CF   50      7D    038E   927                 MOVQ    R0,SELECT_VALUE ; Update the current KEY value
                        0393   928
            50    94    0393   929 NO_MA:          CLRB    R0              ; Indicate the search is to continue
                  05    0395   930                 RSB
                        0396   931
            50    01    90     0396   932 MATCH:          MOVB    #1,R0           ; Indicate search is over
                  05    0399   933                 RSB
                        039A   934
                        039A   935
                        039A   936 ;
                        039A   937 ;   Action routines for comparisons
                        039A   938 ;
                        039A   939 COMPARE_ACT:                                 ;
      10 63       0E    E0     039A   940                 BBS     #CNR$V_SEM_RT,(R3),20$ ; If action routine, call it now
                  08    ED     039E   941                 CMPZV   #CNR$V_SEM_TYP,- ; If data resides in bitmask in CNF,
00          63    03           03A0   942                 #CNR$S_SEM_TYP,(R3),#CNR$C_SEM_BIT
                  0C    13     03A3   943                 BEQL    30$             ; then skip the following.  else,
51          54    5A    C1     03A5   944                 ADDL3   R10,R4,R1       ; Get address of descriptor
            51    61    D0     03A9   945                 MOVL    (R1),R1         ; Pick up a longword of data
                  03    11     03AC   946                 BRB     30$
                02C7    30     03AE   947 20$:     BSBW    GET_RT_FIELD                    ; Else go get the info, return with:
                        03B1   948                                                     ;  R1 = address of longword str desc,
                        03B1   949                                                     ;       or binary value
                        03B1   950                                                     ;  R0 = LBS if and only if success
   57 18 AA     55    E1     03B1   951 30$:     BBC     R5,CNF$L_MASK(R10),210$ ; Br if field is invalid
                  08    EF     03B6   952                 EXTZV   #CNR$V_SEM_TYP,- ; Get parameter type
    7E    63    03           03B8   953                 #CNR$S_SEM_TYP,(R3),-(SP)
                        03BB   954                 $DISPATCH  (SP)+,TYPE=L,<-      ; Dispatch by paramater type
                        03BB   955
                        03BB   956                         <CNR$C_SEM_B,   100$>,- ; Byte
                        03BB   957                         <CNR$C_SEM_W,   110$>,- ; Word
                        03BB   958                         <CNR$C_SEM_L,   150$>,- ; Longword
                        03BB   959                         <CNR$C_SEM_BIT, 130$>,- ; Bit
                        03BB   960                         <CNR$C_SEM_STR, 160$>,- ; String descriptor
                        03BB   961                 >
                        03C9   962                 BUG_CHECK NETNOSTATE,FATAL      ; Type is undefined
                        03CD   963
            51    51    9A     03CD   964 100$:    MOVZBL  R1,R1                   ; Get field
                  15    11     03D0   965                 BRB     150$            ;
            51    51    3C     03D2   966 110$:    MOVZWL  R1,R1                   ; Get field
                  10    11     03D5   967                 BRB     150$            ;
      07 63       0E    E1     03D7   968 130$:    BBC     #CNR$V_SEM_RT,(R3),140$ ; Br if "real" CNF field
51    51    01    00    EF     03DB   969                 EXTZV   #0,#1,R1,R1     ; Else get low bit of value setup by
                        03E0   970                                                 ; action routine
                  05    11     03E0   971                 BRB     150$            ; Continue
51    6A    01    54    EF     03E2   972 140$:    EXTZV   R4,#1,(R10),R1          ; Get the bit value
            51    58    D1     03E7   973 150$:    CMPL    R8,R1                   ; Setup condition codes
                  20    11     03EA   974                 BRB     200$            ; Dispatch
                        03EC   975
      04 63       0E    E0     03EC   976 160$:    BBS     #CNR$V_SEM_RT,(R3),165$ ; If real string,
      51    54    5A    C1     03F0   977                 ADDL3   R10,R4,R1        ; Get address of descriptor in CNF
```

```
                                03F4   978 165$:   PUSHQ   R2                          ; Save regs
                          50 02 A1 3C 03F7   979           MOVZWL  STR_LNG(R1),R0      ; Get string length
                             52 61 3C 03FB   980           MOVZWL  STR_OFF(R1),R2      ; Get offset to string
                             51 52 C0 03FE   981           ADDL    R2,R1               ; Get string pointer
                                   0401   982           PUSHQ   R0                  ; Save descriptor
                61 50 00 68 57 2D 0404   983           CMPC5   R7,(R8),#0,R0,(R1)  ; Setup condition codes
                             0F BA 040A   984           POPR    #^M<R0,R1,R2,R3>    ; Doesn't affect condition codes
                             05 040C   985 200$:   RSB
                                040D   986
                                040D   987 210$:   CLRBIT  #0,R0                   ; Indicate no match
                             8E D5 0411   988           TSTL    (SP)+               ; Pop caller's address
                             05 0413   989           RSB                         ; Return to caller's caller
```

```
                     0414   991          .SBTTL  CNF$GET_FIELD - Get field from CNF entry
                     0414   992 ;+
                     0414   993 ; CNF$GET_FLD_EX - External get zero extended value or descriptor of CNF field
                     0414   994 ; CNF$GET_FIELD  - Internal get zero extended value or descriptor of CNF field
                     0414   995 ;
                     0414   996 ; INPUTS:        R11     Address of CNR
                     0414   997 ;                 R10     Address of CNF
                     0414   998 ;                 R9      FLD # in bits 0:15, Mask I.D. in bits 16:23
                     0414   999 ;                 R0      Error code to be returned if field not active
                     0414  1000 ;
                     0414  1001 ; OUTPUTS:       R9      Unmodified
                     0414  1002 ;                 R8      Parameter value if type bit, byte, word, or longword
                     0414  1003 ;                         Pointer to string if type string
                     0414  1004 ;                 R7      Size of string if type string
                     0414  1005 ;                 R0      Low bit set if field was active
                     0414  1006 ;                         Unchanged otherwise (0 if entered with LBS)
                     0414  1007 ;
                     0414  1008 ;                 NOTE:   R7 and R8 are zeroed at the start of the
                     0414  1009 ;                         routine.  If the routine returns with LBC in R0
                     0414  1010 ;                         then R7 and R8 will equal zero implying a null
                     0414  1011 ;                         field.
                     0414  1012 ;-
                     0414  1013 CNF$GET_FLD_EX::                            ; Get CNF field
0000'CF     DD       0414  1014          PUSHL   NET$GL_FLAGS               ; Save current flags
                     0418  1015          CLRBIT  NET$V_INTRNL,NET$GL_FLAGS ; Indicate external access rights
      50    D4       041E  1016          CLRL    R0                        ; No pre-set error code
      0A    11       0420  1017          BRB     GETFLD                    ; Continue
                     0422  1018
                     0422  1019 CNF$GET_FIELD::                            ; Get CNF field
0000'CF     DD       0422  1020          PUSHL   NET$GL_FLAGS               ; Save current flags
                     0426  1021          SETBIT  NET$V_INTRNL,NET$GL_FLAGS ; Indicate internal access rights
                     042C  1022
                     042C  1023 GETFLD: SETBIT  NET$V_READ,NET$GL_FLAGS   ; Indicate read access intended
   02 50     E9      0432  1024          BLBC    R0,10$                    ; Br if valid error code
      50     D4      0435  1025          CLRL    R0                        ; Else make it valid
      3F     BB      0437  1026 10$:     PUSHR   #^M<R0,R1,R2,R3,R4,R5>    ; Save regs
      57     7C      0439  1027          CLRQ    R7                        ; Zero value/descriptor
    0298     30      043B  1028          BSBW    GET_DSC                   ; Get description of field
   02 50     E9      043E  1029          BLBC    R0,40$                    ; If LBC then no field
      12     10      0441  1030          BSBB    GET                       ; Get the field value
   04 50     E8      0443  1031 40$:     BLBS    R0,50$                    ; If LBS then success
      6E     D5      0446  1032          TSTL    (SP)                      ; Has caller pre-set the error code?
      03     12      0448  1033          BNEQ    60$                       ; If NEQ then yes
6E    50     3C      044A  1034 50$:     MOVZWL  R0,(SP)                   ; Reset the return status
      3F     BA      044D  1035 60$:     POPR    #^M<R0,R1,R2,R3,R4,R5>    ; Restore regs, restore R0
0000'CF 8ED0         044F  1036          POPL    NET$GL_FLAGS              ; Restore flags
            05       0454  1037          RSB
                     0455  1038
                     0455  1039 ;
                     0455  1040 ;  Get Field action routines
                     0455  1041 ;
  10 63     0E       E0    0455  1042 GET:     BBS     #CNR$V_SEM_RT,(R3),10$    ; If action routine, call it now
            08       ED    0459  1043          CMPZV   #CNR$V_SEM_TYP,-          ; If data resides in bitmask in CNF,
00    63    03             045B  1044                  #CNR$S_SEM_TYP,(R3),#CNR$C_SEM_BIT
            0C       13    045E  1045          BEQL    20$                       ; Then skip the following.  else,
51    54    5A       C1    0460  1046          ADDL3   R10,R4,R1                 ; Get pointer to parameter
      51    61       D0    0464  1047          MOVL    (R1),R1                   ; Get a longword of data from CNF
```

```
                     03   11  0467  1048              BRB     20$
                   020C   30  0469  1049  10$:        BSBW    GET_RT_FIELD                ; Else go get the info, return with:
                              046C  1050                                                  ;  R1 = address of longword str desc
                              046C  1051                                                  ;       or binary value
                              046C  1052                                                  ;  R0 = LBS if and only if success
           4E 18 AA   55   E1  046C  1053  20$:        BBC     R5,CNF$L_MASK(R10),170$     ; Br if CNF field is invalid
                      08   EF  0471  1054              EXTZV   #CNR$V_SEM_TYP,-            ; Get parameter type
            7E   63   03        0473  1055                      #CNR$S_SEM_TYP,(R3),-(SP)
                              0476  1056              $DISPATCH  (SP)+,TYPE=L,<-          ; Dispatch by paramater type
                              0476  1057
                              0476  1058                      <CNR$C_SEM_BIT, 100$>,-     ; Bit
                              0476  1059                      <CNR$C_SEM_B,   110$>,-     ; Byte
                              0476  1060                      <CNR$C_SEM_W,   120$>,-     ; Word
                              0476  1061                      <CNR$C_SEM_L,   140$>,-     ; Longword
                              0476  1062                      <CNR$C_SEM_STR, 130$>,-     ; String descriptor
                              0476  1063              >
                              0484  1064              BUG_CHECK NETNOSTATE,FATAL          ; Bug if type is unknown
                              0488  1065
              07 63   0E   E1  0488  1066  100$:      BBC     #CNR$V_SEM_RT,(R3),105$     ; Br if "real" CNF field
        5B    51   01   00   EF  048C  1067              EXTZV   #0,#1,R1,R8                 ; Else get low bit of value setup by
                              0491  1068                                                  ; action routine
                      28   11  0491  1069              BRB     150$                        ; Continue
        5B    6A   01   54   EF  0493  1070  105$:      EXTZV   R4,#1,(R10),R8              ; Get the bit value
                      21   11  0498  1071              BRB     150$
                   58   51   9A  049A  1072  110$:      MOVZBL  R1,R8                       ; Get byte parameter
                      1C   11  049D  1073              BRB     150$
                   58   51   3C  049F  1074  120$:      MOVZWL  R1,R8                       ; Get word parameter
                      17   11  04A2  1075              BRB     150$
                   58   51   D0  04A4  1076  140$:      MOVL    R1,R8                       ; Get longword parameter
                      12   11  04A7  1077              BRB     150$
                18 63   0E   E0  04A9  1078  130$:      BBS     #CNR$V_SEM_RT,(R3),180$     ; Br if the string was obtained from
        51      54   5A   C1  04AD  1079              ADDL3   R10,R4,R1                    ;   an action routine
                58   61   3C  04B1  1080              MOVZWL  STR_OFF(R1),R8              ; Get offset to string
                58   51   C0  04B4  1081              ADDL    R1,R8                       ; Get pointer to string
           57     02 A1   3C  04B7  1082              MOVZWL  STR_LNG(R1),R7              ; Get size of string
                50   01   90  04BB  1083  150$:      MOVB    #1,R0                       ; Indicate field is valid
                         05  04BE  1084  160$:      RSB
                              04BF  1085  170$:      CLRBIT  #0,R0                       ; Indicate field is invalid
                      F9   11  04C3  1086              BRB     160$                        ; And leave
                              04C5  1087
                              04C5  1088              ;
                              04C5  1089              ;  The string was obtained from an action routine and is hence sitting
                              04C5  1090              ;  in the common action routine buffer.  Since this buffer is in
                              04C5  1091              ;  jeapordy of being re-used, it is necessary to allocate a temporary
                              04C5  1092              ;  buffer and move the string to it.  This buffer is inserted on the
                              04C5  1093              ;  NET$GQ_TMP_BUF queue -- all buffers on this queue are deallocated
                              04C5  1094              ;  eventually by one of the higher level routines.
                              04C5  1095              ;
                57   51   D0  04C5  1096  180$:      MOVL    R1,R7                       ; Copy the string descriptor address
           51     02 A1   3C  04C8  1097              MOVZWL  STR_LNG(R1),R1              ; Get the string length
                51   0C   C0  04CC  1098              ADDL    #12,R1                      ; Copy size of buffer header
                   FB2E'  30  04CF  1099              BSBW    NET$ALLOCATE                ; Allocate the buffer from the ACP pool
                      21   50  E9  04D2  1100              BLBC    R0,200$                     ; Br on error
        0000'DF       62   0E  04D5  1101              INSQUE  (R2),@NET$GQ_TMP_BUF        ; Insert buffer on tmp_buf queue.
          08 A2       51   B0  04DA  1102              MOVW    R1,CNR$W_SIZE(R2)           ; Store size for deallocation.
                52    0C   C0  04DE  1103              ADDL    #12,R2                      ; Point to string storage area
                58    52   D0  04E1  1104              MOVL    R2,R8                       ; Make copy for return
```

```
                51   67   3C  04E4  1105            MOVZWL  STR_OFF(R7),R1        ; Get self-relative offset
                51   57   C0  04E7  1106            ADDL    R7,R1                 ; Make it a pointer
             57   02   A7   3C  04EA  1107            MOVZWL  STR_LNG(R7),R7        ; Get size for return
          68    61   57   28  04EE  1108            MOVC3   R7,(R1),(R8)          ; Move the string
                50   01   D0  04F2  1109            MOVL    #1,R0                 ; Set success
                          05  04F5  1110 190$:       RSB
                              04F6  1111
                     57   7C  04F6  1112 200$:       CLRQ    R7                    ; Zero R7, R8 on error
                     FB   11  04F8  1113            BRB     190$                  ; And exit
                              04FA  1114
```

```
                                04FA  1116          .SBTTL  CNF$PUT_FIELD - Store field into CNF entry
                                04FA  1117 ;+
                                04FA  1118 ; CNF$PUT_FLD_EX - External insert CNF field
                                04FA  1119 ; CNF$PUT_FIELD  - Internal insert CNF field
                                04FA  1120 ;
                                04FA  1121 ;
                                04FA  1122 ; INPUTS:        R11     Address of CNR
                                04FA  1123 ;                R10     Address of CNF
                                04FA  1124 ;                R9      FLD # in bits 0:15, Mask I.D. in bits 16:23
                                04FA  1125 ;                R8      Parameter value if type byte, word, or longword
                                04FA  1126 ;                        Pointer to string if type string
                                04FA  1127 ;                R7      Size of string if type string
                                04FA  1128 ;                R0      Error code to be returned upon failure
                                04FA  1129 ;
                                04FA  1130 ; OUTPUTS:       R0      Low bit set if successful
                                04FA  1131 ;                        Unchanged otherwise (0 if entered with LBS)
                                04FA  1132 ;
                                04FA  1133 ;-
                                04FA  1134 CNF$PUT_FLD_EX::                             ; Store CNF field
0000'CF   DD                    04FA  1135          PUSHL   NET$GL_FLAGS                ; Save current flags
                                04FE  1136          CLRBIT  NET$V_INTRNL,NET$GL_FLAGS ; Indicate external access
      0F   11                   0504  1137          BRB     PUTFLD_1                    ; No pre-set error code
                                0506  1138
                                0506  1139 CNF$PUT_FIELD::                            ; Store CNF field
0000'CF   DD                    0506  1140          PUSHL   NET$GL_FLAGS                ; Save current flags
                                050A  1141          SETBIT  NET$V_INTRNL,NET$GL_FLAGS ; Indicate external access
   02 50   E9                   0510  1142          BLBC    R0,PUTFLD_1                 ; Br if valid error code
      50   D4                   0513  1143 PUTFLD: CLRL    R0                          ; No pre-set error code
                                0515  1144 PUTFLD_1:
                                0515  1145          CLRBIT  NET$V_READ,NET$GL_FLAGS ; Indicate write access
      3F   BB                   051B  1146          PUSHR   #^M<R0,R1,R2,R3,R4,R5>      ; Save regs
    01B9   30                   051D  1147          BSBW    GET_DSC                     ; Get description of field
   02 50   E9                   0520  1148          BLBC    R0,40$                      ; If LBC then no field
      12   10                   0523  1149          BSBB    PUT                         ; Store the field
   04 50   E8                   0525  1150 40$:     BLBS    R0,50$                      ; If LBS then success
      6E   D5                   0528  1151          TSTL    (SP)                        ; Has caller pre-set the error code?
      03   12                   052A  1152          BNEQ    60$                         ; If NEQ then yes
   6E   50   3C                 052C  1153 50$:     MOVZWL  R0,(SP)                     ; Reset the return status
      3F   BA                   052F  1154 60$:     POPR    #^M<R0,R1,R2,R3,R4,R5>      ; Restore regs, restore R0
0000'CF BED0                    0531  1155          POPL    NET$GL_FLAGS                ; Restore flags
            05                  0536  1156          RSB
                                0537  1157 ;
                                0537  1158 ;  Put Field action routines
                                0537  1159 ;
                                0537  1160 PUT:
      08   EF                   0537  1161          EXTZV   #CNR$V_SEM_TYP,-            ; Get parameter type
50   63   03                    0539  1162                  #CNR$S_SEM_TYP,(R3),R0      ;
      04   50   D1               053C  1163          CMPL    R0,#CNR$C_SEM_STR           ; String?
           12   12               053F  1164          BNEQ    50$                         ; If NEQ no, br to check value range
           10   ED               0541  1165          CMPZV   #CNR$V_SEM_SMX,-            ; Range check required?
00   63   0C                    0543  1166                  #CNR$S_SEM_SMX,(R3),#0      ;
           07   13               0546  1167          BEQL    40$                         ; If EQL then no
           10   ED               0548  1168          CMPZV   #CNR$V_SEM_SMX,-            ; String length within range?
57   63   0C                    054A  1169                  #CNR$S_SEM_SMX,(R3),R7      ;
           1F   1F               054D  1170          BLSSU   80$                         ; If LSSU then out of range
           57   D5               054F  1171 40$:     TSTL    R7                          ; Is string null?
           15   11               0551  1172          BRB     70$                         ; Continue in commone
```

```
                50   03   D1   0553  1173 50$:    CMPL    #CNR$C_SEM_L,R0          ; Longword value ?
                     0E   13   0556  1174         BEQL    60$                      ; If EQL skip range check
                     10   ED   0558  1175         CMPZV   #CNR$V_SEM_MAX,-         ; Range check required?
          00    63   10         055A  1176                 #CNR$S_SEM_MAX,(R3),#0   ;
                     07   13   055D  1177         BEQL    60$                      ; If EQL then no
                     10   ED   055F  1178         CMPZV   #CNR$V_SEM_MAX,-         ; Within range?
          58    63   10         0561  1179                 #CNR$S_SEM_MAX,(R3),R8   ;
                     08   1F   0564  1180         BLSSU   80$                      ; If LSSU then param value too large
                     58   D5   0566  1181 60$:    TSTL    R8                       ; Is the value zero ?
                     0A   12   0568  1182 70$:    BNEQ    90$                      ; If not continue
             06 63   0F   E0   056A  1183         BBS     #CNR$V_SEM_Z,(R3),90$    ; If BS then zero is okay
          50 0000'8F   3C   056E  1184 80$:    MOVZWL  #SS$_BADPARAM,R0         ; Indicate bad parameter value
                     05   0573  1185         RSB                              ; Return status in R0
                          0574  1186
          51   54   5A   C1   0574  1187 90$:    ADDL3   R10,R4,R1                ; Get pointer to parameter
             08 63   0E   E1   0578  1188         BBC     #CNR$V_SEM_RT,(R3),95$   ; Br if not action routine
                   0148   30   057C  1189         BSBW    PUT_RT_FIELD             ; Call action routine
                      3A 50   E9   057F  1190         BLBC    R0,T70$                  ; If error, do not mark as "set"
                     30   11   0582  1191         BRB     150$                     ; Else, mark as "set" and exit
                          0584  1192
                          0584  1193 95$:    $DISPATCH  R0,<-                 ; Dispatch by paramater type
                          0584  1194
                          0584  1195              <CNR$C_SEM_BIT, 100$>,-     ; Bit
                          0584  1196              <CNR$C_SEM_B,   110$>,-     ; Byte
                          0584  1197              <CNR$C_SEM_W,   120$>,-     ; Word
                          0584  1198              <CNR$C_SEM_L,   130$>,-     ; Longword
                          0584  1199              <CNR$C_SEM_STR, 140$>,-     ; String descriptor
                          0584  1200         >
                          0592  1201         BUG_CHECK NETNOSTATE,FATAL       ; Bug if type is unknown
          51   5A   C2   0596  1202 100$:    SUBL    R10,R1                   ; Subtract out CNF address
    6A 01 51   58   F0   0599  1203         INSV    R8,R1,#1,(R10)           ; Insert bit value
                     14   11   059E  1204         BRB     150$                     ;
                61   58   90   05A0  1205 110$:    MOVB    R8,(R1)                  ; Insert byte parameter
                     0F   11   05A3  1206         BRB     150$                     ;
                61   58   B0   05A5  1207 120$:    MOVW    R8,(R1)                  ; Insert word parameter
                     0A   11   05A8  1208         BRB     150$                     ;
                61   58   D0   05AA  1209 130$:    MOVL    R8,(R1)                  ; Insert longword parameter
                     05   11   05AD  1210         BRB     150$                     ;
                     0C   10   05AF  1211 140$:    BSBB    PUT_STR                  ; Insert the string
                   08 50   E9   05B1  1212         BLBC    R0,T70$                  ; If LBC then didn't fit
                50   01   90   05B4  1213 150$:    MOVB    #1,R0                    ; Indicate success
          00 18 AA   55   E2   05B7  1214         BBSS    R5,CNF$L_MASK(R10),170$  ; Mark field valid
                     05   05BC  1215 170$:    RSB
                          05BD  1216
                          05BD  1217
                          05BD  1218 PUT_STR:                                 ; Insert string into CNF block
                          05BD  1219         ;
                          05BD  1220         ;  If the new string is less than or equal to the size of the new
                          05BD  1221         ;  string, then simply re-use the space.  This is needed to make
                          05BD  1222         ;  is simple to store fixed size strings, such as NI addresses,
                          05BD  1223         ;  without having to generate a new CNF block, when the SIZ_FREE
                          05BD  1224         ;  is exhausted.  Any waste holes for unequal strings will be wasted.
                          05BD  1225         ;
                          05BD  1226         ;  If string is already active then subtract its size from
                          05BD  1227         ;  CNF$W_SIZ_USED before storing the string.  Store the string and
                          05BD  1228         ;  update CNF$W_SIZ_USED and CNF$W_SIZ_FREE to account for storage
                          05BD  1229         ;  taken.
```

```
                                05BD  1230          ;
                                05BD  1231          ;   INPUTS:     R10 = CNF block pointer
                                05BD  1232          ;               R8  = Pointer to string
                                05BD  1233          ;               R7  = Length of string
                                05BD  1234          ;               R5  = Bit offset from CNF mask to field active flag
                                05BD  1235          ;               R1  = Address of CNF string descriptor
                                05BD  1236          ;               R0  = Scratch
                                05BD  1237          ;
                                05BD  1238          ;   OUTPUTS:    R1  = Garbage
                                05BD  1239          ;               R0  = SS$_NORMAL  if successful
                                05BD  1240          ;                     SS$_INSFMEM otherwise
                                05BD  1241          ;
                     3C   BB    05BD  1242          PUSHR   #^M<R2,R3,R4,R5>        ; Save regs
                                05BF  1243          ;
                                05BF  1244          ;   If the new string is less than, or equal to, the size of the
                                05BF  1245          ;   original string, then simply re-use its space (wasting any
                                05BF  1246          ;   excess), and modify the length of the parameter.  This is done
                                05BF  1247          ;   to make replacement of fixed size strings easy.
                                05BF  1248          ;
           17 18 AA    55   E1  05BF  1249          BBC     R5,CNF$L_MASK(R10),20$  ; If BC then field currently inactive
              02 A1    57   B1  05C4  1250          CMPW    R7,STR_LNG(R1)          ; Equal or less space than original?
                        11   1A  05C8  1251          BGTRU   20$                     ; If not, then allocate new space
           50   02 A1    57   A3  05CA  1252          SUBW3   R7,STR_LNG(R1),R0       ; Compute difference in sizes
                10 AA    50   A2  05CF  1253          SUBW    R0,CNF$W_SIZ_USED(R10)  ; Adjust string space taken
                   53    61   3C  05D3  1254          MOVZWL  STR_OFF(R1),R3          ; Get offset to original string
                   53    51   C0  05D6  1255          ADDL    R1,R3                   ; Get pointer to string space
                         2F   11  05D9  1256          BRB     50$                     ; Move the string, and exit
                                05DB  1257          ;
                                05DB  1258          ;   We cannot re-use the space of the original string.  Deallocate
                                05DB  1259          ;   the space used by the original string, if any (wasting it), and
                                05DB  1260          ;   allocate some new space at the end of the block.
                                05DB  1261          ;
     50    0000'8F    3C  05DB  1262  20$:    MOVZWL  #SS$_INSFMEM,R0         ; Assume no space left
        0E AA    57   B1  05E0  1263          CMPW    R7,CNF$W_SIZ_FREE(R10)  ; Enough free space left ?
                  2F   1A  05E4  1264          BGTRU   90$                     ; If GTRU then no
        53    0C AA    9E  05E6  1265          MOVAB   CNF$W_OFF_FREE(R10),R3  ; Prepare to calc. ptr
              52   63    3C  05EA  1266          MOVZWL  (R3),R2                 ; Get offset to free space
              53   52    C0  05ED  1267          ADDL2   R2,R3                   ; Calculate ptr to free space
                         05F0  1268          ASSUME  STR_OFF EQ 0
     61       53   51    A3  05F0  1269          SUBW3   R1,R3,STR_OFF(R1)       ; Enter self-relative offset
  05 18 AA    55   E1  05F4  1270          BBC     R5,CNF$L_MASK(R10),30$  ; If BC then field currently inactive
                02 A1    A2  05F9  1271          SUBW    STR_LNG(R1),-           ; Adjust space used (note that we are
                   10 AA         05FC  1272                      CNF$W_SIZ_USED(R10); return it to CNF$W_SIZ_FREE)
        0E AA    57   A2  05FE  1273  30$:    SUBW    R7,CNF$W_SIZ_FREE(R10)  ; Account for space taken
        10 AA    57   A0  0602  1274          ADDW    R7,CNF$W_SIZ_USED(R10)  ; Account for space taken
        0C AA    57   A0  0606  1275          ADDW    R7,CNF$W_OFF_FREE(R10)  ; Advance free space offset
        02 A1    57   B0  060A  1276  50$:    MOVW    R7,STR_LNG(R1)          ; Enter string size
     63    68    57   28  060E  1277          MOVC3   R7,(R8),(R3)            ; Move it
              50   00'   D0  0612  1278          MOVL    S^#SS$_NORMAL,R0        ; Indicate success
                   3C   BA  0615  1279  90$:    POPR    #^M<R2,R3,R4,R5>        ; Restore regs
                        05   0617  1280          RSB
```

```
                        0618  1282          .SBTTL  CNF$CLR_FIELD - Clear a CNF field
                        0618  1283 ;+
                        0618  1284 ; CNF$CLR_FLD_EX - External clear CNF field
                        0618  1285 ; CNF$CLR_FIELD  - Internal clear CNF field
                        0618  1286 ;
                        0618  1287 ; INPUTS:        R11     CNR pointer
                        0618  1288 ;                R10     CNF pointer (CNF$CLEAR only)
                        0618  1289 ;                R9      Field f.d.
                        0618  1290 ;
                        0618  1291 ; OUTPUTS:       R0      LBS if successful, LBC otherwise
                        0618  1292 ;
                        0618  1293 ;                All other registers are preserved.
                        0618  1294 ;-
                        0618  1295 CNF$CLR_FLD_EX::                             ; Clear bit in CNF mask
          0000'CF   DD  0618  1296          PUSHL   NET$GL_FLAGS                ; Save current flags
                        061C  1297          CLRBIT  NET$V_INTRNL,NET$GL_FLAGS ; Indicate external access
                 0A   11  0622  1298          BRB     CLRFLD
                        0624  1299
                        0624  1300 CNF$CLR_FIELD::                              ; Clear CNF field
          0000'CF   DD  0624  1301          PUSHL   NET$GL_FLAGS                ; Save current flags
                        0628  1302          SETBIT  NET$V_INTRNL,NET$GL_FLAGS ; Indicate external access
                        062E  1303
                        062E  1304 CLRFLD: CLRBIT  NET$V_READ,NET$GL_FLAGS   ; Indicate write access
            02 50   E9  0634  1305          BLBC    R0,5$                       ; Br if valid error code
               50   D4  0637  1306          CLRL    R0                          ; Else make it valid
               3F   BB  0639  1307 5$:      PUSHR   #^M<R0,R1,R2,R3,R4,R5>      ; Save regs
             009B   30  063B  1308          BSBW    GET_DSC                     ; Get field semantics
            1D 50   E9  063E  1309          BLBC    R0,10$                      ; Br if not defined
      1B 18 AA      55  E5  0641  1310          BBCC    R5,CNF$L_MASK(R10),10$  ; Clear the bit
         14 63      0E  E0  0646  1311          BBS     #CNR$V_SEM_RT,(R3),10$  ; Br if "field" is an action routine
                    08  ED  064A  1312          CMPZV   #CNR$V_SEM_TYP,-        ; Is this a string field ?
               63   03      064C  1313                  #CNR$S_SEM_TYP,(R3),-
                    04      064E  1314                  #CNR$C_SEM_STR
                    0D  12  064F  1315          BNEQ    10$                     ; If NEQ no, we're done
                    00  EF  0651  1316          EXTZV   #CNR$V_SEM_OFF,-        ; Get offset from top of CNF to field
         52         63  08      0653  1317                  #CNR$S_SEM_OFF,(R3),R2
               52   52  5A  C0  0656  1318          ADDL    R10,R2              ; Make it a pointer
                  02   A2  A2  0659  1319          SUBW    STR_LNG(R2),-       ; Update amount of space used
                  10   AA      065C  1320                  CNF$W_SIZ_USED(R10)
                  04   50  E8  065E  1321 10$:     BLBS    R0,20$              ; If LBS then success
                       6E  D5  0661  1322          TSTL    (SP)                ; Has caller pre-set the error code?
                       03  12  0663  1323          BNEQ    30$                 ; If NEQ then yes
               6E      50  3C  0665  1324 20$:     MOVZWL  R0,(SP)             ; Reset the return status
                       3F  BA  0668  1325 30$:     POPR    #^M<R0,R1,R2,R3,R4,R5>  ; Restore regs
                0000'CF    8ED0  066A  1326          POPL    NET$GL_FLAGS        ; Restore flags
                           05  066F  1327          RSB
```

```
                     0670  1329          .SBTTL  CNF$VERIFY - Check if field exists
                     0670  1330 ;+
                     0670  1331 ; CNF$VERIFY  - See if field semantics are defined
                     0670  1332 ;
                     0670  1333 ; INPUTS:       R11     CNR pointer
                     0670  1334 ;               R10     CNF pointer
                     0670  1335 ;               R9      Field i.d.
                     0670  1336 ;
                     0670  1337 ; OUTPUTS:      R0      LBS if successful, LBC otherwise
                     0670  1338 ;
                     0670  1339 ;               All other registers are preserved.
                     0670  1340 ;-
                     0670  1341 CNF$VERIFY::                                ; Are field semantics defined?
  3E   BB            0670  1342          PUSHR   #^M<R1,R2,R3,R4,R5>       ; Save critical regs
 00BC  30            0672  1343          BSBW    GET_DSC_1                 ; Get field semantics
  3E   BA            0675  1344 10$:     POPR    #^M<R1,R2,R3,R4,R5>       ; Restore regs
  05   05            0677  1345          RSB
```

```
                                                .SBTTL  GET_RT_FIELD - Call action routine to get value
                                        ;+
                                        ; GET_RT_FIELD - Call action routine to get a parameter value
                                        ;
                                        ; Inputs:
                                        ;
                                        ;       R11 = Address of CNR
                                        ;       R10 = Address of CNF
                                        ;       R9 = Field ID
                                        ;       R5 = Bit offset from top of CNF mask vector to field presence flag
                                        ;       R4 = Address of action routine
                                        ;       R3 = Address of field semantics longword
                                        ;
                                        ; Outputs:
                                        ;
                                        ;       R0 = Status code
                                        ;       R1 = Address of longword "field value"
                                        ;               For binary values, longword binary value
                                        ;               For string values, address of word offset & word count
                                        ;
                                        ;       R2-R11 are preserved.
                                        ;
                                        ; The action routine is called with the following interface:
                                        ;
                                        ; Input to action routine:
                                        ;
                                        ;       R0 = 0, indicating parameter is to be read, not written.
                                        ;               (used only for those action routines that can do both).
                                        ;       R11 = Address of CNR
                                        ;       R10 = Address of CNF
                                        ;       R3 = Address of scratch buffer
                                        ;
                                        ; Output from action routine:
                                        ;
                                        ;       For string values, R3 points just beyond string in scratch buffer.
                                        ;       For binary values, R1 contains the value itself.
                                        ;
                                        ;       All registers (R2-R11) can be destroyed by action routine before
                                        ;       returning here.
                                        ;-
                                        ;
                                        GET_RT_FIELD:
                OFFC 8F     BB          PUSHR   #^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Save registers
                     08     ED          CMPZV   #CNR$V_SEM_TYP,-                ; String value?
          04    63   03                        #CNR$S_SEM_TYP,(R3),#CNR$C_SEM_STR
                     06     13          BEQL    50$                             ; Branch if so
                                        ;
                                        ; Call action routine for binary value
                                        ;
                                        ;
                     50     D4          CLRL    R0                              ; Indicate parameter to be read
                     64     16          JSB     (R4)                            ; Call action routine
                     2F     11          BRB     90$                             ; Return status in R0
                                        ;
                                        ;
```

```
                                0689  1404 ; Call action routines for string
                                0689  1405 ;
                                0689  1406
      34 000C'CF    01    E2    0689  1407 50$:    BBSS    #TMP_V_BUF,TMP_B_FLAGS,100$ ; Allocate static buffer
      53   00000004'GF    9E    068F  1408         MOVAB   G^TMP_BUF,R3                ; Setup buffer pointer
                   50    D4    0696  1409         CLRL    R0                          ; Indicate parameter to be read
                   64    16    0698  1410         JSB     (R4)                        ; Call action routine
      51   00000000'GF    9E    069A  1411         MOVAB   G^TMP_VAL,R1                ; Point to descriptor storage
      52   00000004'GF    9E    06A1  1412         MOVAB   G^TMP_BUF,R2                ; Get original pointer
      02 A1    53    52    A3    06A8  1413         SUBW3   R2,R3,STR_LNG(R1)           ; Setup string size
         61    0004'8F    B0    06AD  1414         MOVW    #TMP_BUF-TMP_VAL,STR_OFF(R1) ; Setup string offset
      0B 000C'CF    01    E5    06B2  1415         BBCC    #TMP_V_BUF,TMP_B_FLAGS,100$ ; Deallocate static buffer
                                0688  1416
               0FFC 8F    BA    0688  1417 90$:    POPR    #^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Restore registers
   18 AA    01    55    50    F0    068C  1418         INSV    R0,R5,#1,CNF$L_MASK(R10); Remember validity of field
                              05    06C2  1419         RSB                         ; Return status in R0
                                06C3  1420
                                06C3  1421 100$:   BUG_CHECK       NETNOSTATE,FATAL
```

```
                     06C7  1423          .SBTTL  PUT_RT_FIELD - Call action routine to store value
                     06C7  1424 ;+
                     06C7  1425 ; PUT_RT_FIELD - Call action routine to store a parameter value
                     06C7  1426 ;
                     06C7  1427 ; Inputs:
                     06C7  1428 ;
                     06C7  1429 ;       R11 = Address of CNR
                     06C7  1430 ;       R10 = Address of CNF
                     06C7  1431 ;       R9 = Field ID
                     06C7  1432 ;       R7/R8 = Parameter value
                     06C7  1433 ;       R5 = Bit offset from top of CNF mask vector to field presence flag
                     06C7  1434 ;       R4 = Address of action routine
                     06C7  1435 ;       R3 = Address of field semantics longword
                     06C7  1436 ;
                     06C7  1437 ; Outputs:
                     06C7  1438 ;
                     06C7  1439 ;       R0 = Status code
                     06C7  1440 ;
                     06C7  1441 ;       R2-R11 are preserved.
                     06C7  1442 ;
                     06C7  1443 ;
                     06C7  1444 ; The action routine is called with the following interface:
                     06C7  1445 ;
                     06C7  1446 ; Input to action routine:
                     06C7  1447 ;
                     06C7  1448 ;       R0 = 1, indicating parameter is to be written, not read.
                     06C7  1449 ;            (used only for those action routines that can do both).
                     06C7  1450 ;       R11 = Address of CNR
                     06C7  1451 ;       R10 = Address of CNF
                     06C7  1452 ;       R7/R8 = Parameter value (descriptor if string, else R8 = longword).
                     06C7  1453 ;
                     06C7  1454 ; Output from action routine:
                     06C7  1455 ;
                     06C7  1456 ;       R0 = True if parameter was stored, else false.
                     06C7  1457 ;
                     06C7  1458 ;       All registers (R2-R11) can be destroyed by action routine before
                     06C7  1459 ;       returning here.
                     06C7  1460 ;-
                     06C7  1461
                     06C7  1462 PUT_RT_FIELD:
OFFC 8F     BB       06C7  1463          PUSHR   #^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Save registers
50     01   D0       06CB  1464          MOVL    #1,R0                       ; Indicate parameter to be written
       64   16       06CE  1465          JSB     (R4)                        ; Call action routine
OFFC 8F     BA       06D0  1466          POPR    #^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Restore registers
       05   05       06D4  1467          RSB                                 ; Return status in R0
                     06D5  1468
                     06D5  1469 100$:    BUG_CHECK       NETNOSTATE,FATAL
```

```
                                06D9  1471          .SBTTL  GET_DSC - Get descriptor of CNF field
                                06D9  1472 ;+
                                06D9  1473 ; GET_DSC       - Get descriptor of CNF field and check access rights
                                06D9  1474 ; GET_DSC_1     - Get descriptor of CNF field
                                06D9  1475 ;
                                06D9  1476 ; inputs:       R11  Address of CNR
                                06D9  1477 ;               R9   FLD number in bits 0-15, mask id in bits 16-23
                                06D9  1478 ;
                                06D9  1479 ; outputs:      R11  Address of CNR
                                06D9  1480 ;               R9   Unmodified
                                06D9  1481 ;               R5   Bit offset from top of CNF mask vector to bit in R9
                                06D9  1482 ;               R4   Byte offset from top of CNF to parameter or
                                06D9  1483 ;                    pointer to action routine (depending upon semanitics)
                                06D9  1484 ;               R3   Address of field semantics longword
                                06D9  1485 ;               R0   LBS if successful
                                06D9  1486 ;                    LBC otherwise
                                06D9  1487 ;-
                                06D9  1488 GET_DSC:                                     ; Get descriptor and check access rights
                         56   10  06D9  1489          BSBB    GET_DSC_1               ; Get the descriptor
                      4C 50   E9  06DB  1490          BLBC    R0,50$                  ; Br on error
                         0B   EF  06DE  1491          EXTZV   #CNR$V_SEM_ACC,-        ; Get access protection
                50   63   03      06E0  1492                  #CNR$S_SEM_ACC,(R3),R0  ;
                         0A   E0  06E3  1493          BBS     #NET$V_READ,-           ; Br if read access is intended
                   22 0000'CF      06E5  1494                  NET$GL_FLAGS,20$
                                06E9  1495          ;
                                06E9  1496          ;   Write access is intended.  The boolean equation for NOT allowing
                                06E9  1497          ;   write access is:
                                06E9  1498          ;                               -W = R0 + (ER+NE)*(-INTRNL) + CW*LOCKED
                                06E9  1499          ;
                      01   50   91  06E9  1500          CMPB    R0,#CNR$C_ACC_RO        ; Read only ?
                           3D   13  06EC  1501          BEQL    60$                     ; If EQL no access permitted
                      04   50   91  06EE  1502          CMPB    R0,#CNR$C_ACC_ER        ; External read only ?
                           05   13  06F1  1503          BEQL    8$                      ; If so, then check if external
                      05   50   91  06F3  1504          CMPB    R0,#CNR$C_ACC_NE        ; No external read or write access?
                           06   12  06F6  1505          BNEQ    10$                     ; If not, then continue
                           09   E1  06F8  1506  8$:     BBC     #NET$V_INTRNL,-         ; If BC then not internal access
                   2D 0000'CF      06FA  1507                  NET$GL_FLAGS,60$        ;
                      03   50   91  06FE  1508  10$:    CMPB    R0,#CNR$C_ACC_CW        ; Is field conditionally writeable?
                           1E   12  0701  1509          BNEQ    30$                     ; If NEQ then access is allowed
                           0B   E1  0703  1510          BBC     #NET$V_CNFLCK,-         ; If BC then okay to write the field
                   18 0000'CF      0705  1511                  NET$GL_FLAGS,30$        ;
                           20   11  0709  1512          BRB     60$                     ; Else cannot write it
                                070B  1513          ;
                                070B  1514          ;   Read access intended.  The boolean equation for allowable read
                                070B  1515          ;   access is:
                                070B  1516          ;                       R = -(NE*-INTRNL) * (-WO + WO*INTRNL + WO*BYPASS)
                                070B  1517          ;
                           09   E0  070B  1518  20$:     BBS     #NET$V_INTRNL,-         ; Br if internally accessed
                   10 0000'CF      070D  1519                  NET$GL_FLAGS,30$        ;
                      05   50   91  0711  1520          CMPB    R0,#CNR$C_ACC_NE        ; No external read/write access?
                           15   13  0714  1521          BEQL    60$                     ; If not, then disallow access
                           08   E0  0716  1522          BBS     #NET$V_BYPASS,-         ; Br if user has bypass privilege
                   05 0000'CF      0718  1523                  NET$GL_FLAGS,30$        ;
                      02   50   91  071C  1524          CMPB    R0,#CNR$C_ACC_WO        ; Is field "write-only"
                           04   13  071F  1525          BEQL    40$                     ; If EQL then no access allowed
                      50   01   90  0721  1526  30$:     MOVB    #1,R0                   ; Set success
                                05  0724  1527          RSB                             ;
```

```
                                0725  1528
                                0725  1529 40$:    MOVZWL  #SS$_BADPARAM,R0        ; No read access allowed
                                05    072A  1530 50$:    RSB                             ;
                                      072B  1531
                                3C    072B  1532 60$:    MOVZWL  #SS$_WRITLCK,R0         ; No write access allowed
                                05    0730  1533          RSB                             ;
                                      0731  1534
                                      0731  1535
                                      0731  1536
                                      0731  1537 GET_DSC_1:
                                9A    0731  1538          MOVZBL  CNR$B_TYPE(R11),R0      ; Get database i.d.
                             50 59  08 18  ED  0735  1539          CMPZV   #NFB$V_DB,#NFB$S_DB,R9,R0 ; Is if for this database ?
                                18    24  12  073A  1540          BNEQ    40$             ; If NEQ then no
                                      073C  1541
                                      073C  1542          ASSUME  NFB$V_INX  EQ  0
                                      073C  1543          ASSUME  NFB$S_INX  EQ  16
                                      073C  1544
                                59    3C    073C  1545          MOVZWL  R9,R5           ; Get field index
                             0E AB    55    D1  073F  1546          CMPL    R5,CNR$W_MAX_INX(R11)   ; Is it within range ?
                                1B    1A  0743  1547          BGTRU   40$             ; If GTRU then out of range
                          53  0128 CB45    DE  0745  1548          MOVAL   CNR$L_SEM_TAB(R11)[R5],R3 ; Point to semantic longword
                                00    EF  074B  1549          EXTZV   #CNR$V_SEM_OFF,-        ; Get byte offset to field from
                             54 63    08        074D  1550                  #CNR$S_SEM_OFF,(R3),R4  ; top of CNF (or routine index)
                                0E    13  0750  1551          BEQL    40$             ; Branch if no semantic entry
                             06 63    0E  E1  0752  1552          BBC     #CNR$V_SEM_RT,(R3),30$  ; Br if "field" is not a routine
                                54    5B  C0  0756  1553          ADDL    R11,R4          ; Get address of pointer to routine
                                54    64  D0  0759  1554          MOVL    (R4),R4         ; Get address of routine
                                50    00' D0  075C  1555 30$:    MOVL    S^#SS$_NORMAL,R0        ; Indicate success
                                05    075F  1556          RSB
                                      0760  1557
                                50    0000'8F   3C  0760  1558 40$:    MOVZWL  #SS$_BADPARAM,R0        ; Indicate illegal field ID
                                05    0765  1559          RSB
                                      0766  1560
                                      0766  1561
                                      0766  1562 .END
```

```
ACP$C_STA_F                     = 00000004
ACP$C_STA_H                     = 00000005
ACP$C_STA_I                     = 00000000
ACP$C_STA_N                     = 00000001
ACP$C_STA_R                     = 00000002
ACP$C_STA_S                     = 00000003
BIT...                          = 00000006
BUG$_NETNOSTATE                   ********    X   05
CLRFLD                            0000062E R      05
CNF$B_FLG                       = 0000000B
CNF$B_TYPE                      = 0000000A
CNF$CLONE                         000001A0 RG     05
CNF$CLR_FIELD                     00000624 RG     05
CNF$CLR_FLD_EX                    00000618 RG     05
CNF$COPY                          00000170 RG     05
CNF$DELETE                        00000015 RG     05
CNF$GET_FIELD                     00000422 RG     05
CNF$GET_FLD_EX                    00000414 RG     05
CNF$INIT                          00000234 RG     05
CNF$INIT_UTL                      0000021D RG     05
CNF$INSERT                        00000044 RG     05
CNF$KEY_SEARCH                    0000026A RG     05
CNF$KEY_SRCH_EX                   00000258 RG     05
CNF$L_MASK                      = 00000018
CNF$M_FLG_ACP                   = 00000004
CNF$M_FLG_CNR                   = 00000001
CNF$M_FLG_DELETE                = 00000002
CNF$PRE_QIO                       00000009 RG     05
CNF$PRE_SHOW                      00000000 RG     05
CNF$PURGE                         00000040 RG     05
CNF$PUT_FIELD                     00000506 RG     05
CNF$PUT_FLD_EX                    000004FA RG     05
CNF$SEARCH                        00000288 RG     05
CNF$SEARCH_EX                     0000027C RG     05
CNF$VERIFY                        00000670 RG     05
CNF$V_FLG_ACP                   = 00000002
CNF$V_FLG_DELETE                = 00000001
CNF$W_ID                        = 00000012
CNF$W_OFF_FREE                  = 0000000C
CNF$W_SIZE                      = 00000008
CNF$W_SIZ_FREE                  = 0000000E
CNF$W_SIZ_USED                  = 00000010
CNF$_ADVANCE                    = 00000000
CNF$_QUIT                       = 00000002
CNF$_TAKE_CURR                  = 00000003
CNF$_TAKE_PREV                  = 00000001
CNR$B_TYPE                      = 0000000A
CNR$C_ACC_CW                    = 00000003
CNR$C_ACC_ER                    = 00000004
CNR$C_ACC_NE                    = 00000005
CNR$C_ACC_RO                    = 00000001
CNR$C_ACC_WO                    = 00000002
CNR$C_MAX_INX                   = 0000005F
CNR$C_SEM_B                     = 00000001
CNR$C_SEM_BIT                   = 00000000
CNR$C_SEM_L                     = 00000003
CNR$C_SEM_STR                   = 00000004
CNR$C_SEM_W                     = 00000002
CNR$L_ACT_DELETE                = 00000028
CNR$L_ACT_DFLT                  = 00000020
CNR$L_ACT_INSERT                = 00000024
CNR$L_ACT_QIO                   = 00000018
CNR$L_ACT_REMOVE                = 0000002C
CNR$L_ACT_SHOW                  = 0000001C
CNR$L_INSERT                    = 00000034
CNR$L_SCANNER                   = 00000030
CNR$L_SEM_TAB                   = 00000128
CNR$L_SPCSCAN                   = 00000038
CNR$L_VEC_MAND                  = 00000080
CNR$L_VEC_UNIQ                  = 000000E4
CNR$S_SEM_ACC                   = 00000003
CNR$S_SEM_MAX                   = 00000010
CNR$S_SEM_OFF                   = 00000008
CNR$S_SEM_SMX                   = 0000000C
CNR$S_SEM_TYP                   = 00000003
CNR$V_SEM_ACC                   = 00000008
CNR$V_SEM_MAX                   = 00000010
CNR$V_SEM_OFF                   = 00000000
CNR$V_SEM_RT                    = 0000000E
CNR$V_SEM_SMX                   = 00000010
CNR$V_SEM_TYP                   = 00000008
CNR$V_SEM_Z                     = 0000000F
CNR$W_MAX_INX                   = 0000000E
CNR$W_SIZE                      = 00000008
CNR$W_SIZ_CNF                   = 0000000C
COMPARE                           00000347 R      05
COMPARE_ACT                       0000039A R      05
DLIST                           = 00000004
DYN$C_NET                       = 00000017
GET                               00000455 R      05
GETFLD                            0000042C R      05
GET_DSC                           000006D9 R      05
GET_DSC_1                         00000731 R      05
GET_RT_FIELD                      00000678 R      05
KEY_EQL                           0000035D R      05
KEY_GTRU                          00000369 R      05
KEY_LSSU                          0000036F R      05
KEY_MAX                           00000375 R      05
KEY_MIN                           00000380 R      05
KEY_NEQ                           00000363 R      05
MATCH                             00000396 R      05
NET$ALLOCATE                      ********    X   05
NET$C_ACT_TIMER                 = 0000001E
NET$C_EFN_ASYN                  = 00000002
NET$C_EFN_WAIT                  = 00000001
NET$C_IPL                       = 00000008
NET$C_MAXACCFLD                 = 00000027
NET$C_MAXLINNAM                 = 0000000F
NET$C_MAXLNK                    = 000003FF
NET$C_MAXNODNAM                 = 00000006
NET$C_MAXOBJNAM                 = 0000000C
NET$C_MAX_AREAS                 = 0000003F
NET$C_MAX_LINES                 = 00000040
NET$C_MAX_NCB                   = 0000006E
```

```
NET$C_MAX_NODES                 = 000003FF
NET$C_MAX_OBJ                   = 000000FF
NET$C_MAX_WQE                   = 00000014
NET$C_MINBUFSIZ                 = 000000C0
NET$C_TID_ACT                   = 00000003
NET$C_TID_RUS                   = 00000001
NET$C_TID_XRT                   = 00000002
NET$C_TRCTL_CEL                 = 00000002
NET$C_TRCTL_OVR                 = 00000005
NET$C_UTLBUFSIZ                 = 00001000
NET$GL_FLAGS                      ********  X   05
NET$GL_UTLBUF                     ********  X   05
NET$GQ_TMP_BUF                    ********  X   05
NET$M_MAXLNKMSK                 = 000003FF
NET$V_BYPASS                    = 00000008
NET$V_CNFLCK                    = 0000000B
NET$V_INTRNL                    = 00000009
NET$V_PURGE                     = 0000000E
NET$V_READ                      = 0000000A
NFB$C_OP_EQL                    = 00000000
NFB$C_OP_FNDMAX                 = 00000005
NFB$C_OP_FNDMIN                 = 00000004
NFB$C_OP_FNDPOS                 = 00000006
NFB$C_OP_GTRU                   = 00000001
NFB$C_OP_LSSU                   = 00000002
NFB$C_OP_NEQ                    = 00000003
NFB$C_WILDCARD                  = 00000001
NFB$S_DB                        = 00000008
NFB$S_INX                       = 00000010
NFB$V_DB                      o = 00000018
NFB$V_INX                       = 00000000
NO_MA                             00000393  R       05
NSP$C_EXT_LNK                   = 0000001E
NSP$C_MAXADR                    = 00000009
PUT                               00000537  R       05
PUTFLD                            00000513  R       05
PUTFLD_1                          00000515  R       05
PUT_RT_FIELD                      000006C7  R       05
PUT_STR                           000005BD  R       05
SCAN                              00000129  R       05
SEARCH                            00000292  R       05
SELECT_CNF                        00000000  R       03
SELECT_VALUE                      00000004  R       03
SIZ...                          = 00000001
SLIST                           = 00000008
SPCSCAN                           0000010E  R       05
SS$_BADPARAM                      ********      X   05
SS$_DEVACTIVE                     ********      X   05
SS$_ENDOFFILE                     ********      X   05
SS$_INSFARG                       ********      X   05
SS$_INSFMEM                       ********      X   05
SS$_NORMAL                        ********      X   05
SS$_WRITLCK                       ********      X   05
STR_LNG                         = 00000002
STR_OFF                         = 00000000
TMPBUF_DESC                       00000000  RG      02
TMP_BUF                           00000004  R       04
TMP_BUF_END                       00000450  R       04
TMP_B_FLAGS                       0000000C  R       03
TMP_LTH                         = 0000044C
TMP_VAL                           00000000  R       04
TMP_V_BUF                       = 00000001
TMP_V_VAL                       = 00000000
TR$C_MAXHDR                     = 0000001C
TR$C_NI_ALLEND1                 = 040000AB
TR$C_NI_ALLEND2                 = 00000000
TR$C_NI_ALLROU1                 = 030000AB
TR$C_NI_ALLROU2                 = 00000000
TR$C_NI_PREFIX                  = 000400AA
TR$C_NI_PROT                    = 00000360
TR$C_PRI_ECL                    = 0000001F
TR$C_PRI_RTHRU                  = 0000001F
UPD                               00000389  R       05
_$$_                            = 000000EF
```

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
ABS	00000000 (0.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
NET_PURE	00000008 (8.)	02 (2.)	NOPIC USR CON REL LCL NOSHR NOEXE RD NOWRT NOVEC LONG
NET_IMPURE	0000000D (13.)	03 (3.)	NOPIC USR CON REL LCL NOSHR NOEXE RD WRT NOVEC BYTE
TABLES_IMPURE	00000454 (1108.)	04 (4.)	NOPIC USR CON REL GBL NOSHR NOEXE RD WRT NOVEC BYTE
NET_CODE	00000766 (1894.)	05 (5.)	NOPIC USR CON REL LCL NOSHR EXE RD NOWRT NOVEC BYTE

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	28	00:00:00.08	00:00:00.48
Command processing	131	00:00:00.97	00:00:03.24
Pass 1	428	00:00:14.07	00:00:22.44
Symbol table sort	0	00:00:01.29	00:00:01.42
Pass 2	291	00:00:04.13	00:00:05.64
Symbol table output	23	00:00:00.18	00:00:00.18
Psect synopsis output	3	00:00:00.04	00:00:00.05
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	906	00:00:20.77	00:00:33.46

The working set limit was 2000 pages.
75794 bytes (149 pages) of virtual memory were used to buffer the intermediate code.
There were 60 pages of symbol table space allocated to hold 879 non-local and 131 local symbols.
1562 source lines were read in Pass 1, producing 27 object records in Pass 2.
29 pages of virtual memory were used to define 25 macros.

Macro library statistics

Macro library name	Macros defined
_255DUA28:[SHRLIB]NMALIBRY.MLB;1	0
_255DUA28:[SHRLIB]EVCDEF.MLB;1	0
_255DUA28:[NETACP.OBJ]NETDRV.MLB;1	0
_255DUA28:[NETACP.OBJ]NET.MLB;1	8
_255DUA28:[SYS.OBJ]LIB.MLB;1	2
_255DUA28:[SYSLIB]STARLET.MLB;2	6
TOTALS (all libraries)	16

1008 GETS were required to define 16 macros.

There were no errors, warnings or information messages.

```
MACRO/LIS=LIS$:NETCNF/OBJ=OBJ$:NETCNF MSRC$:NETCNF/UPDATE=(ENH$:NETCNF)+EXECML$/LIB+LIB$:NET/LIB+LIB$:NETDRV/LIB+SHRLIB$:EVCDEF/LIB+
```

NETCNF
LIS

NETCNFACT
LIS

NETCNFDLL
LIS